Complex additive geometric multilevel solvers for Helmholtz equations on spacetrees

BRAM REPS, University of Antwerp
TOBIAS WEINZIERL, Durham University

We introduce a family of implementations of low order, additive, geometric multilevel solvers for systems of Helmholtz equations arising from Schrödinger equations. Both grid spacing and arithmetics may comprise complex numbers and we thus can apply complex scaling to the indefinite Helmholtz operator. Our implementations are based upon the notion of a spacetree and work exclusively with a finite number of precomputed local element matrices. They are globally matrix-free.

Combining various relaxation factors with two grid transfer operators allows us to switch from additive multigrid over a hierarchical basis method into a Bramble-Pasciak-Xu (BPX)-type solver, with several multiscale smoothing variants within one code base. Pipelining allows us to realise full approximation storage (FAS) within the additive environment where, amortised, each grid vertex carrying degrees of freedom is read/written only once per iteration. The codes realise a single-touch policy. Among the features facilitated by matrix-free FAS is arbitrary dynamic mesh refinement (AMR) for all solver variants. AMR as enabler for full multigrid (FMG) cycling—the grid unfolds throughout the computation—allows us to reduce the cost per unknown per order of accuracy.

The present paper primary contributes towards software realisation and design questions. Our experiments show that the consolidation of single-touch FAS, dynamic AMR and vectorisation-friendly, complex scaled, matrix-free FMG cycles delivers a mature implementation blueprint for solvers of Helmholtz equations in general. For this blueprint, we put particular emphasis on a strict implementation formalism as well as some implementation correctness proofs.

CCS Concepts: •**Mathematics of computing** → Mathematical software; •**Computing methodologies** → Parallel computing methodologies;

Additional Key Words and Phrases: Helmholtz, additive multigrid, BPX, AMR, vectorisation

ACM Reference Format:
Bram Reps, Tobias Weinzierl, 2015. Complex additive geometric multilevel solvers for Helmholtz equations on spacetrees. *ACM Trans. Math. Softw.* V, N, Article A (January YYYY), 36 pages.
DOI: 0000001.0000001

1. INTRODUCTION

The present paper's efforts are driven by an interest in the dynamics of p quantum particles, originally described by the time-dependent Schrödinger wave equation [Schrödinger 1926]. The wave equation's operator is the sum of the kinetic energies of the individual particles and their potential energy determined by their nature and interaction. The *Hamiltonian* operator drives the complex-valued time derivative. Numerical methods solving Schrödinger equations are of great interest for the simulation and prediction of the reaction rates of fundamental processes in few-body physics and chemistry [Faddeev and Merkuriev 1993; Plasma 2010 Committe 2007; Raizer 1991; Smirnov 2015; Vanroose et al. 2005]. The breakup of the H_2-molecule with $p = 2$ parti-

Author's addresses: B. Reps, Department of Mathematics and Computer Science, University of Antwerp, 2020 Antwerp, Belgium; T. Weinzierl, School of Engineering and Computing Sciences, Durham University, DH1 3LE Durham, United Kingdom

DOI: 0000001.0000001

cles for example is solved in [Vanroose et al. 2005]. A key ingredient there is the expansion of the time-dependent Schrödinger equation into a time-independent Schrödinger equation being a $d = 3p$-dimensional Helmholtz problem. This ansatz allows a post-processing step to determine the *far field map* (FFM) which yields the probability distribution of particles escaping. It is written down as function of the angle from the point of interaction and is calculated by integrating over stationary Helmholtz solutions (Figure 1). Since the inversion of the arising matrices becomes unfeasible for growing p, we require robust and fast iterative solvers.

To achieve this, the present paper follows the aforementioned work [Vanroose et al. 2005]. It uses a partial wave expansion to decompose the time-independent $d = 3p$-dimensional system further into a cascade of p-dimensional Helmholtz problems. In essence, this tackles the problem with a transformed basis, measuring distances between free and stationary particles. They are referred to as *channels*. The base expansion is truncated, i.e. we focus only on the dominant channels, and we end up with an iterative scheme where a set of uncoupled Helmholtz problems has to be solved within the iterative loop as preconditioner. This yields perfectly parallel (embarrassingly parallel [Foster 1995]) channel solves. Further, we rely on the fact that FFM integrals can be calculated on a complex contour instead of on the original real domain [Cools et al. 2014a]. This implies that we may solve Helmholtz equations which are rotated in the complex domain. These equations are better posed while still yielding high quality FFM results. A robust, fast solver thus has to perform particularly good on the lowest level of parallelisation, i.e. exploit vector units, where the perfect parallelism does not pay off directly.

We propose to realise a complex-valued, matrix-free, additive multilevel solver for the present challenge where combinations of well-suited relaxation parameters and grid transfer operators allow us to apply an additive multigrid solver, a hierarchical basis approach or a Bramble-Pasciak-Xu (BPX)-type solver [Bramble et al. 1990]. Within one code base, we may choose a stable solver depending on the equations' characteristics. It combines the idea of additive multigrid [Bastian et al. 1998] with full approximation storage (FAS) based upon a hierarchical generating system [Griebel 1990] which resembles the MLAT idea [Brandt 1973; 1977], and it realises all data structures within a p-dimensional spacetree traversed by a depth-first search automaton [Weinzierl 2009; Weinzierl and Mehl 2011]. Our Helmholtz solver supports dynamically adaptive meshes resolving localised wave characteristics. Furthermore, its in-situ meshing allows us to unfold the grid on-the-fly similar to FMG cycles in the multiplicative context. In our preconditioning environment, multiple problem parameter choices finally are fused into one grid sweep as long as it increases the arithmetic intensity.

The novel algorithmic contributions of the present paper span algorithms, application-specific experience and a challenging application: First, we integrate various sophisticated multigrid techniques concisely into one code base such that we can offer the additive multilevel solvers with a one-touch policy. Each vertex is, on average, loaded into the caches only once per iteration and resides inside the caches only briefly. All ingredients are vertically, i.e. between the grid levels, and horizontally, i.e. in the grid, integrated. Similar techniques have been proposed for multilevel solvers [Adams et al. 2016; Mehl et al. 2006; Ghysels et al. 2012; Ghysels and Vanroose 2015] or Krylov solvers [Chronopoulos and Gear 1989; Hoemmen 2010; Ghysels et al. 2013; Ghysels and Vanroose 2014], but, to the best of our knowledge, no other approach offers a solution representation on all levels plus single touch. Multilevel solution representations simplify the handling of hanging nodes, non-linear problems and scale-dependent discretisations [Cools et al. 2014b]. Second, we show that this realisation embeds into a depth-first traversal of a tree spanning the cascade of grids embedded into each

other. Such trees—octrees, quadtrees and variations of those—are popular in many application areas. They allow us to realise arbitrary dynamic refinement and coarsening in combination with in-situ mesh generation that seamlessly integrate into the stream-like data processing. There is no significant setup cost for the meshing, while the memory footprint is minimalist as we work matrix-freely and encode the underlying grid structure with only few bits per cell [Weinzierl and Mehl 2011]. Simple data flow analysis reveals that such a merger of FAS with the ideas of additive multigrid and spacetrees is not straightforward if data is not processed multiple times. Third, we show that the cheap dynamic adaptivity allows us to tailor the grid to the solution characteristics. Furthermore, if the solver runs a cascade of additive cycles over finer and finer grids, the resulting scheme mirrors FMG where coarse solves act as initial guess for finer grids. It is able to reduce, for benchmark problems, the residual by one order of magnitude every 1.5 traversals. This is remarkable given that we use merely a Jacobi smoother and geometric inter-grid transfer operators. Fourth, we report on reasonable MFlop rates and vectorisation efficiency as we fuse the solution of multiple preconditioning problems into one adaptive grid; despite the fact that we employ a low order scheme which is notoriously bandwidth bound and do not rely on sophisticated smoother optimisation techniques [Kowarschik et al. 2000]. This renders the algorithmic mindset well-suited for upcoming machine generations that are expected to obtain a significant part of their capability from vectorisation's extreme concurrency in combination with constrained memory [Dongarra et al. 2014]. Fifth, we demonstrate how complex scaling and various choices of relaxation and very few operators allow a user to obtain a set of solvers that can be tailored to many problems. Notably, we can robustly solve four-dimensional Helmholtz problems which is a significant improvement over previous work. Finally, all algorithmic steps are presented in a compact form and all ingredients come along with correctness proofs. While linear algebra packages supporting complex arithmetics per se are rare, a rigorous formal description enables reprogramming and reuse for different applications.

We identify four major limitations of the present work. First, we do not offer a strategy to tackle the curse of dimensionality [Bellman 1961] spelled through p rising. Though we show that the FMG-type cycles reduce the cost per unknown per accuracy by magnitudes, such a reduction of cost, even in combination with the vectorisation and memory access results, is far below what is required to tackle large p. In practice, we are still bound to $p \leq 4$ though the implementation would support arbitrary big p. Second, we do not pick up the discussion on well-suited smoothers for the present problems. We show that our solvers achieve robustness due to the complex rotations. However, their efficiency deteriorates. We emphasise that the efficient nature of the present implementation patterns makes us hope that they can be used as starting point to realise more competitive smoothers as proposed in [Chen et al. 2012; Ernst and Gander 2011; Ghysels et al. 2012; Ghysels and Vanroose 2015; Stolk 2015], e.g. Yet, this is future work. Third, we do not realise problem-dependent grid transfer or coarse grid operators. Such operators are mandatory to tackle problems with spatially varying PDE properties as they occur for matching boundary conditions, e.g., within the multigrid setting. For the present case studies, our solvers' efficiency here suffers. We refer to promising tests with BoxMG within the spacetree paradigm [Weinzierl 2013; Yavneh and Weinzierl 2012] for pathways towards future work. Finally, any multiplicative considerations are out of scope here. All shortcomings highlight that the present paper primary contributes towards software realisation and design questions. We also focus on single node performance as our algorithmic framework is perfectly parallel. This neither implies that the presented approach can not be tuned further with respect to parallelism nor do we address related challenges such as proper load balancing. Notably, techniques such as segmental refinement [Adams et al. 2016] that

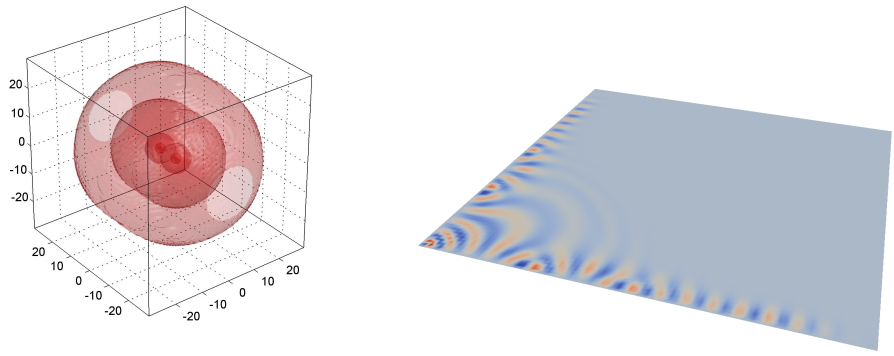

Fig. 1. Left: Potential field defined by two fixed (heavy) particles. A three-dimensional Helmholtz equation (1) describes the behaviour, i.e. probability distribution, of a single free particle within this field. Applications are interested in its far field map (FFM) which is a volume integral over Helmholtz solutions within a sphere enclosing the setup—typically for $p > 1$. Right: Solution of one channel's Helmholtz problem of type (3) for a $p = 2$ setup. The bottom and left axis encode the distance from the $p = 2$ centers of mass. The domain expands infinitely into the right and top direction.

spatially decompose fine grid solves could help to increase the concurrency while preserving the present work's vertical and horizontal integration as well as the fusion of parameter spaces.

The remainder is organised as follows: We detail the physical context and solver framework in Section 2 before we introduce our notion of spacetrees yielding the multiscale grid (Section 3). Starting from a recapitulation of standard additive multigrid, we introduce our particular single-touch multilevel ingredients in Section 4. They are fused into one additive scheme with different flavours in Section 5. We next give some correctness proofs for the realisation (Section 6). Some numerical results in Section 7 precede a brief conclusion and an outlook.

2. APPLICATION CONTEXT

A time-dependent Schrödinger equation for p free particles can be solved by projecting the initial state ($t = 0$) onto the Hamiltonian's eigenstates. Each quantum eigenstate Ψ parameterised in spherical coordinates r_j around the particles is factorisable as

$$\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_p, t) \equiv e^{-\imath Et} \psi^{3p}(\mathbf{r}_1, \ldots, \mathbf{r}_p),$$

where E is the eigenvalue, that is the eigenstate's total energy. As the probability distribution $|\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_p, t)|^2 = |\psi^{3p}(\mathbf{r}_1, \ldots, \mathbf{r}_p)|^2$ is constant in time [Cohen-Tannoudji et al. 1977], these modes are stationary states. Substituting a stationary state into the time-dependent Schrödinger equations yields a $3p$-dimensional time-independent Schrödinger equation, that is a Helmholtz equation, of the form

$$\begin{aligned} H_{3p}\psi^{3p}(\mathbf{r}_1, \ldots, \mathbf{r}_p) &\equiv \left[-\Delta_{3p} - \phi^{3p}(\mathbf{r}_1, \ldots, \mathbf{r}_p)\right] \psi^{3p}(\mathbf{r}_1, \ldots, \mathbf{r}_p) \\ &= \chi^{3p}(\mathbf{r}_1, \ldots, \mathbf{r}_p), \end{aligned} \tag{1}$$

where ϕ^{3p} and χ^{3p} depend on the setup's configuration comprising also the impact of additional fixed (heavy) particles as well as the free particles' properties. Usually solely long-term steady state solutions of the particle quantum system can be measured, so (1) needs only be solved once for the known total energy E of the system.

Although the time-dependence is removed from the governing equation, the dimensionality still grows large with the number of particles. Following [Baertschy and Li 2001], the multi-channel approach [Vanroose et al. 2005] expresses the $3p$-dimensional

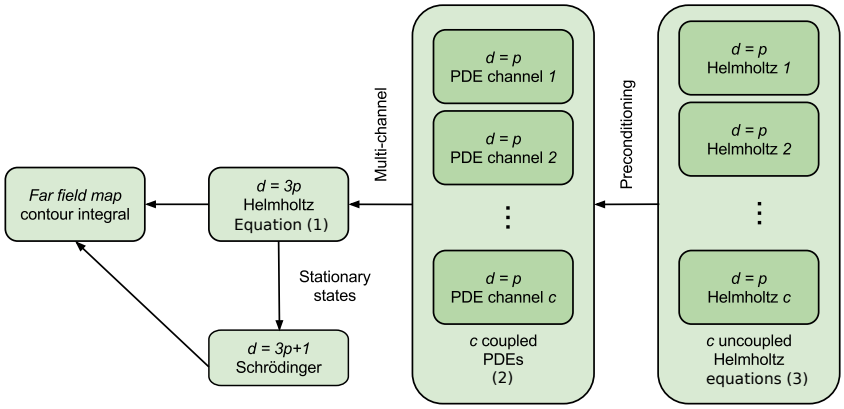

Fig. 2. Visualisation of the overall workflow from [Vanroose et al. 2005] to determine the far field map of a quantum mechanical scattering problem. The workflow circumnavigates the Schrödinger equation where the $+1$ dimension represents the time. We aim at tackling the set of c Helmholtz problems of dimension $d = p$ for $p > 2$.

solution of (1) in terms of partial waves. It rewrites ψ^{3p} as a sum of projections onto partial waves (*channels*) in decreasing order of magnitude and truncates this sum after $c \in \mathbb{N}_0$ terms. This involves a transformation of $\mathbf{r}_j$ into a radial distance and solid angle. It yields c coupled PDEs

$$\begin{pmatrix} H_{11} & A_{12} & \dots & A_{1c} \\ A_{21} & H_{22} & & A_{2c} \\ \vdots & & \ddots & \vdots \\ A_{c1} & A_{c2} & \dots & H_{cc} \end{pmatrix} \begin{pmatrix} \psi_1^{mc} \\ \psi_2^{mc} \\ \vdots \\ \psi_c^{mc} \end{pmatrix} = \begin{pmatrix} \chi_1^{mc} \\ \chi_2^{mc} \\ \vdots \\ \chi_c^{mc} \end{pmatrix} \tag{2}$$

with p-dimensional Helmholtz operators $H_{ii} : \mathcal{C}^2(\mathbb{R}) \to \mathcal{C}^2(\mathbb{R})$ on the diagonal. Off-diagonal operators $A_{ij} : \mathcal{C}^2(\mathbb{R}) \to \mathcal{C}^2(\mathbb{R})$ contain potential terms and couple the channels. The closer to the diagonal, the stronger the coupling. We split

$$A \equiv \begin{pmatrix} H_{11} & & & \\ & H_{22} & & \\ & & \ddots & \\ & & & H_{cc} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & A_{12} & \dots & A_{1c} \\ A_{21} & \mathbf{0} & & A_{2c} \\ \vdots & & \ddots & \vdots \\ A_{c1} & A_{c2} & \dots & \mathbf{0} \end{pmatrix},$$

bring the non-diagonal blocks to the right-hand side and end up with an iterative scheme on the block level. Each block row yields a problem of the form

$$[-\Delta_p - \phi^p(\rho_1, \dots, \rho_p)]\, \psi^p(\rho_1, \dots, \rho_p) = \chi^p(\rho_1, \dots, \rho_p). \tag{3}$$

It is solved on the unit hypercube $\rho \in (0,1)^p$ as finite subregion of $(0,\infty)^p$, while the modified right-hand side χ^p anticipates the coupling operators. The technical details of this transformation are given in [bin Zubair et al. 2012]. Whenever we drop the p superscripts from here on, the symbols are generic for any of the channel PDEs.

Each of these equations has to be solved efficiently. Each solve acts as a preconditioning step within the overall algorithm. While the spectrum of Δ_p per Helmholtz operator on the diagonal retains large condition numbers, all system matrices are sparse. A concurrent solve of such channels in a Jacobi-type fashion is perfectly parallel and thus not studied further here. We note that a $p = 2$-dimensional Helmholtz problem is solved successfully with direct methods for the blocks in [Vanroose et al. 2005] using a parallel computer. For $p > 2$, direct solves however are not feasible anymore.

For $\phi^p(\rho_1, \dots, \rho_p) > 0$, the Helmholtz operator can be indefinite, which disturbs the convergence of standard iterative methods [Ernst and Gander 2011]. Among many other publications on the subject, the field of *shifted Laplacian preconditioning* has greatly inspired the solvers in the current paper. The first preconditioners of this kind were the Laplacian and the positively shifted Laplacian introduced in [Bayliss et al. 1983], later generalised to complex-valued shifts [Erlangga et al. 2004; Erlangga et al. 2006]. Alternative preconditioners and solution methods are derived from frequency shift time integration [Meerbergen and Coyette 2009], moving perfectly matched layers [Engquist and Ying 2011], a transformation of the Helmholtz equation to a reaction-advection-diffusion problem [Haber and MacLachlan 2011], separation of variables [Plessix and Mulder 2003], the wave-ray approach [Brandt and Livshits 1997], Krylov subspace methods as smoother substitute [Elman et al. 2001], or algebraic multilevel methods [Bollhöfer et al. 2009; Tsuji and Tuminaro 2015]. This list is not comprehensive.

We use the *Complex Scaled Grid (CSG)* operator [Reps et al. 2010]. It maps (3) onto a complex scaled or rotated domain, i.e. makes $\rho_j \in (0, e^{\imath\theta})$. θ denotes this rotation in the complex plane. We thus solve closely related Helmholtz equations that are better conditioned and still merge into a good block diagonal preconditioner for (2). Near to open domain boundaries, complex-valued scaling introduces absorbing boundary layers that we use in combination with zero-Dirichlet conditions. This complex scaling is chosen independent of parameter ω in the smoother. In fact, technically, CSG broken down to a grid is equivalent to treating the entire domain as an absorbing boundary layer.

Because of the complex domain rotation, the eigenvalues are rotated in the complex plane; away from the origin. Standard multigrid methods thus can be applied, whereas multigrid on an unmodified equation ($\theta \mapsto 0$) fails. The approach is inspired by the *complex shifted Laplacian (CSL)* where complex damping is introduced in the Helmholtz shift, i.e. $\phi \to (1 + \alpha\imath)\phi$ with $0 < \alpha < 1$ [Erlangga et al. 2004; Erlangga et al. 2006]. We refer to [Magolu monga Made et al. 2000; van Gijzen et al. 2007; Osei-Kuffuor and Saad 2010; Cools and Vanroose 2013] and the extensive literature on the CSL operators for a study of the appropriate choice for parameter θ, a discussion that is beyond the scope of the present work. Both CSG acting as preconditioner and the channel decomposition preserve the solution characteristics of the FFM (Figure 2) which is the final quantity of interest, and we conclude that the channel decomposition's reduction of dimensionality of the subproblems starts to pay off for $p \geq 2$.

3. SPACETREES

ALGORITHM 1: Textbook additive multigrid. An iteration is triggered by ADD(ℓ_{max}), i.e. runs from finest to coarsest grid. See Remark 4.1 on exact ℓ_{min} solves.

```
1:  function ADD(ℓ)
2:      r_ℓ ← χ_ℓ − H_ℓ u_ℓ                    ▷ p-linear shape functions determine H.
3:      u_ℓ ← u_ℓ + ω_S S(u_ℓ, χ_ℓ)            ▷ Smoother S with damping ω_S ∈ (0, 1).
4:      if ℓ > ℓ_min then
5:          u_{ℓ−1} ← 0
6:          χ_{ℓ−1} ← R r_ℓ                    ▷ p-linear geometric restriction.
7:          ADD(ℓ − 1)
8:          u_ℓ ← u_ℓ + ω_cg P u_{ℓ−1}         ▷ Coarse grid damping ω_cg ∈ (0, 1).
9:                                             ▷ p-linear prolongation P.
        end
10: end function
```

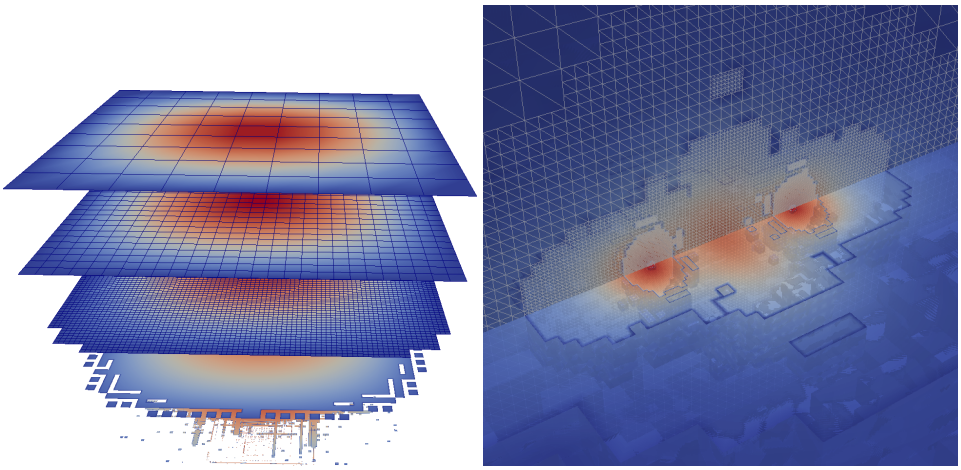

Fig. 3. Left: The spacetree yields a cascade of regular grids. Two dimensional setting with grid levels from top to down. While the union forms an adaptive Cartesian grid and while the grids are geometrically embedded into each other, a grid on a single level might be ragged (finest level). Multiple vertices belonging to different levels coincide spatially, all the vertices on the two coarsest levels left are *refined*. Right: Illustration of heavy hydrogen ${}^2H$ with one free electron and a fixed proton and neutron. Direct solution of (1). The adaptivity yields grids that range over ten levels in the simulation domain.

Our implementation of multilevel solvers such as additive multigrid (Algorithm 1) relies on a finite element formulation where the geometric elements are hypercubes over the complex domain. Their dimension is $3p$ for Helmholtz problems of type (1) and p for Helmholtz problems of type (3) in the channel approach. Except for some illustrations from the application domain, we focus on numerical results for the latter and therefore use p as dimension from hereon. For $p = 2$, we start with the unit square suitably scaled by $e^{\imath\theta} \in \mathbb{C}$. This square is our coarsest grid $\Omega_{h,0}$ holding one cell and 2^p vertices. It coincides with the computational domain. Let its grid entities have level $\ell = 0$. We next split the cube equidistantly into three parts along each coordinate axis and end up with 3^p new squares. They describe a grid $\Omega_{h,1}$ and belong to level $\ell = 1$. Our choice of three-partitioning stems from the fact that we use the software Peano [Weinzierl et al. 2012] for our realisation. All algorithmic ideas work for bi-partitioning as well.

For each of the 3^p squares of level $\ell = 1$ we decide independently whether we refine them further. As we continue recursively, we end up with a cascade of regular grids that might be disconnected (Figure 3 left). The extension of this construction to any $p > 2$ is straightforward (e.g. Figure 3 right). Our overall scheme describes a spacetree [Weinzierl 2009; Weinzierl and Mehl 2011] and is an extension of the classic octree and quadtree idea to three-partitioning in combination with arbitrary spatial dimension d. The spacetree exhibits the following properties that are important to our solvers:

(1) It yields a set of grids where each grid of level ℓ is a refinement of the grid of level $\ell - 1$, i.e. the grids are strictly embedded into each other.

$$\Omega_{h,0} \subset \Omega_{h,1} \subset \ldots \subset \Omega_{h,\ell-1} \subset \Omega_{h,\ell}. \tag{4}$$

(2) The union of all grids yields an adaptive Cartesian grid

$$\Omega_h = \bigcup_{\ell} \Omega_{h,\ell}. \tag{5}$$

(3) A vertex is unique due to its spatial position plus its level, i.e. multiple vertices may coincide on the same position $x \in \Omega$.

(4) The individual grids are regular Cartesian grids aligned to each other. But they can be ragged as not all cells of a level ℓ have to be refined to obtain the grid at level $\ell + 1$. Ω_h comprises hanging nodes.

The equivalence of the cascade of adaptive Cartesian grids with a spacetree is well-known ([Weinzierl 2009] or [Bader 2013] and references therein). It motivates our inverse use of the term level with respect to standard multigrid literature. In this context, we also emphasise that we prefer the term erase as counterpart to refinement, as coarsening has already a semantics in the multigrid context.

Let a cell $a \in \Omega_{h,\ell}$ be a parent of $b \in \Omega_{h,\ell+1}$, if b is constructed from a due to one refinement step. This parent-child relation introduces a partial order on the set of all grid cells of all grid levels. It defines the spacetree. Given a spacetree, any tree traversal is equivalent to a multiscale element-wise grid traversal. Particular advantageous is the combination of space-filling curves (we use the Peano curve here) [Bader 2013], adaptive Cartesian multiscale grids and a depth-first traversal, as it yields memory-efficient codes: Basically, the tree traversal can be mapped onto a depth-first pushback automaton where a child is never visited prior to its parent. As soon as this automaton encounters an unrefined spacetree cell, a leaf, it backtracks the tree and continues to descend within another subtree. Two bits per vertex then are sufficient to encode both adjacency and dynamic adaptivity information [Weinzierl and Mehl 2011]—though we typically use a whole byte to make programming easier—while the whole data structure is linearised on few stacks or streams. We work with a linearised octree [Sundar et al. 2008]. All presented solver ingredients fit also to other tree traversals. Notably, breadth-first of parallel traversals [Weinzierl 2015] do work. We solely require two properties to hold: Parents have to be traversed prior to their children, and any solver has to have the opportunity to plug into both the steps from level ℓ to level $\ell + 1$ and the other way round. The former requirement allows us to realise arbitrary dynamic adaptivity; the tree may unfold throughout the steps down. The latter requirement allows us to distribute algorithmic ingredients both among the unfolding of the traversal and its backtracking. The depth of the backtracking is limited by the maximum tree depth. It is small.

We close our spacetree discussion with a few technical terms. A *hanging vertex* is a vertex with less than 2^p adjacent spacetree cells on the same level. A *refined vertex* is a vertex where all 2^p adjacent spacetree cells on the same level have children (Figure 3). A non-hanging vertex is a *fine grid vertex* if no other non-hanging vertex at the same spatial position with higher level does exist. Let $\mathbb{V}$ be the set of non-hanging vertices in the grid. A fine grid cell is an unrefined spacetree cell. The finest level of the spacetree from here on shall be ℓ_{max}. As adaptive multigrid solves often do not coarsen the problem completely, we rely formally on a coarsest compute level $\ell_{min} \geq 1$, as all vertices on level 0 coincide with the domain boundary and, for Dirichlet boundary conditions, do not carry unknowns.

4. MULTIGRID REALISATION

Our work is based on a Ritz-Galerkin finite element formulation of (3) with p-linear shape functions and a nodal unknown association. Better-suited, problem-tailored discretisations such as dispersion minimising schemes [Chen et al. 2012; Stolk 2015] are beyond scope here but can be realised within our computational framework. All shape functions are centred around vertices, and we make each shape functions cover exactly the 2^p cells of the vertex's level. As the spacetree yields multiple vertices at one space coordinate and as each level ℓ of the grid spans one function space U_ℓ, the whole spacetree induces a hierarchical generating system [Griebel 1994]. Let each $v \in \mathbb{V}$ hold a three-tuple $(u, \phi, \chi)(v) \in \mathbb{C}^3$. u is the weight of the shape function associated to the

vertex, i.e. for shape functions $\psi(v)$ the solution of the discretised problem is given by $\psi = \sum_{v \in \mathbb{V}} u(v)\psi(v)$. According to (3), ϕ holds a weight of the identity discretised by shape and test function space. For a fine grid vertex, χ accordingly holds the weight of the discretised right-hand side. For a refined vertex, χ holds the right-hand side of the multigrid scheme. To reduce notation, we reuse the function symbols u, ϕ and χ from the continuous formulation for the vectors of nodal weights, as the semantics of the symbols is disambiguous, and we omit the v-parameterisation. For all quantities, let the subscript $_\ell$ select levels.

H_ℓ in Algorithm 1 has a two-fold meaning. In unrefined vertices, it is a generic symbol for one Helmholtz operator from (2). It comprises a complex rotation factor and is subject to a smoother S that, in our case, denotes one Jacobi smoothing step

$$S_\ell(u_\ell, \chi_\ell) = diag(H_\ell)^{-1}(\chi_\ell - H_\ell u_\ell).$$

In refined vertices, it encodes a correction equation. On one level ℓ, regions may exist where H_ℓ has either of these two functions, i.e. where fine cells align the edge of a coarser cell.

A generating system in combination with the fact that each multigrid sweep starts from a coarse grid correction being zero renders the realisation of an additive multigrid for a regular grid straightforward: We set $u(v) = 0$ for all refined vertices, and initialise two temporary variables $r(v) = 0$ and $diag(H_\ell)(v) = 0$ everywhere. Once we run into a cell, we evaluate the local system matrix in that cell and accumulate the result within the residual r of the 2^p adjacent vertices. Further, it is convenient to assemble the diagonal element. For plain Dirichlet boundary conditions and rediscretisation, an accumulation of $diag(H_\ell)(v) = 0$ is unnecessary. The value is known explicitly from ϕ. For complex scaling of cells near to the boundary as required for absorbing boundary layers or spatially varying ϕ, such an explicit accumulation of $diag$ is mandatory. Once all adjacent cells of a vertex on one level have been visited, we may add the right-hand side to the residual and update the nodal value accordingly. The right-hand side is not required earlier throughout the solve process. The overall process is detailed in [Mehl et al. 2006] and reiterated in Appendix B. It is completely matrix-free. It never sets up any global matrix. This property also holds for restriction and prolongation which are done throughout the grid/tree traversal process as well.

We use geometric grid transfer operators. P is a p-linear interpolation and $R = P^T$. Since we rely on finite element rediscretisation and a uniform complex rotation on all grid levels, H_ℓ on each level is a linear combination of a Laplace operator and a mass operator. The weighting of the two operators is, besides suitable h-scaling, invariant. It is straightforward to validate that both operator component rediscretisations equal a Galerkin coarse grid operator [Trottenberg et al. 2001] if the complex scaling and ϕ are invariant. We obtain $H_\ell = RH_{\ell+1}P$. As we rely on rediscretisation, $diag(H_\ell)^{-1}$ further can be determined on-the-fly on all levels as we accumulate the residual. For varying ϕ or along boundaries with complex scaling, our rediscretised operator slightly deviates from the Galerkin coarse grid operator. However, it still yields reasonable coarse grid corrections. Hanging vertices are interpolated from the next coarser grid through p-linear interpolation, i.e. their value results from P. We note that we do not impose any balancing condition [Sundar et al. 2008] on the spacetree. With these ingredients, we can interpret the fine grid solve as a domain decomposition approach: each fine grid region of a given resolution is assigned Dirichlet values through its hanging nodes as well as real boundary vertices and computes a local solve. The next multigrid iteration, these Dirichlet values are changed to the most recent, i.e. updated solution on the coarser grids.

Our algorithmic sketch so far tackles the treatment of hanging nodes but omits two major challenges: How do fine grid solutions from a level ℓ interact with fine grid so-

lutions of coarser resolutions, and how do we handle, on any fixed level, the transition of the operator semantics from PDE discretisation into multigrid operator along resolution boundaries? We propose to rely on the concept of MLAT [Brandt 1973; 1977] for hierarchical generating systems which leads into the idea of Griebel's HTMG [Griebel 1990]. To make a function representation unique within the generating system, we enforce

$$u_{\ell-1} = Iu_\ell \tag{6}$$

where I is the injection, i.e. plain copying of u weights within our spacetree. This way, vertices that coincide with non-hanging vertices on finer levels act as Dirichlet points for coarser fine grid problems. Constraint (6) formalises a full approximation storage scheme (FAS) on our hierarchical generating system. Homogeneous Dirichlet boundary conditions imply $u_0 \equiv 0$. Though no equation systems are to be solved on $\ell < \ell_{min}$, we nevertheless define u_ℓ on the coarser levels. This does simplify our formulae and arithmetics, as we rely on the notion of hierarchical surpluses [Griebel 1994] defined by the image of the operator $id - PI$ with id being the identity. We stick to $Hu = \chi$ on the fine grid, but formally split the unknown scaling on refined vertices into current solution plus correction. Following [Griebel 1990], this yields

$$\begin{aligned} H_\ell(u_\ell + e_\ell) &= \chi_\ell + H_\ell u_\ell \\ &= Rr_{\ell+1} + RH_{\ell+1}PIu_{\ell+1} = R(\chi_{\ell+1} - H_{\ell+1}u_{\ell+1} + H_{\ell+1}PIu_{\ell+1}) \\ &= R(\chi_{\ell+1} - H_{\ell+1}\hat{u}_{\ell+1}) =: R\hat{r}_{\ell+1} \qquad \text{with } \hat{u}_\ell := (id - PI)u_\ell. \end{aligned} \tag{7}$$

The hierarchical surplus $\hat{u}$ is easy to compute if the coarse grid already holds the injected solution. Computing $\hat{r}$ parallel to the nodal residual for the smoother is easy as well as it requires the same operations as the original fine grid equation system. The prolongation of the coarse grid correction finally simplifies if we add an updated value on level ℓ to the hierarchical surplus $\hat{u}_{\ell+1}$ to obtain a new nodal solution $u_{\ell+1}$ on level $\ell + 1$:

$$u_\ell \leftarrow u_\ell + Pe_{\ell-1} = \hat{u}_\ell + Pu_{\ell-1}.$$

Such a scheme is agnostic whether original equation (in unrefined vertices) or multigrid correction (in refined vertices) is solved on a level ℓ. All unknowns are of the same scaling and the u weights have the same semantics everywhere. However, it misfits top-down tree traversals.

Remark 4.1. Textbook multigrid classically relies on an exact solve on the coarsest grid. We replace this solve by a sole smoothing step. On the one hand, our experiments demonstrate empirically that this yields sufficiently accurate coarse grid solutions here. On the other hand, our solver acts as preconditioner. Convergence to machine precision is not required.

This remark requires three addenda: First, our additive multigrid solvers damp coarse grid contributions for stability reasons. The impact of the coarsest correction then is small anyway and an exact solve would not pay off. Second, in Poisson-like experiments we coarsen into very small grids (with only 2^p unknowns, e.g.). The small grid problems then are that small that one smoothing step already reduces the residual significantly. Finally, our complex rotations rotate the solution into a Poisson-like regime. Otherwise, the problem could not be represented reasonably on very coarse grids and would require an exact solve.

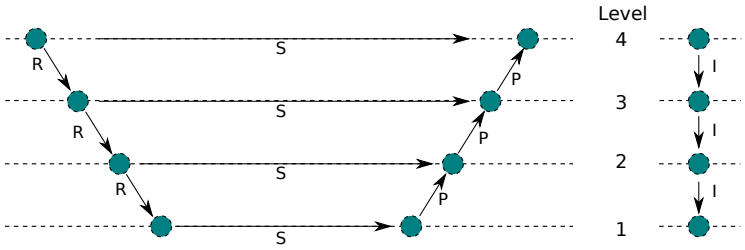

Fig. 4. Data flow diagram for the additive multigrid from Algorithm 1 (left) and data flow diagram for the injection of FAS (right).

5. SINGLE-SWEEP ADDITIVE FAS WITHIN TREE TRAVERSALS

THEOREM 5.1. *Classic additive multigrid in combination with full approximation storage (FAS) can not be implemented with one tree traversal per cycle if we do not use additional variables as well as algorithmic reformulations.*

PROOF. A proof relies on a data flow analysis (Figure 4). The additive multigrid's prolongation prolongs corrections from coarser to finer grids where these are merged with the smoother's impact into an unknown update. Each update requires a data flow to coarser grids due to the injection in (6). The resulting graph has cycles. □

One might argue that a temporary violation of (6) on coarse grids is acceptable. Yet,

(1) our notions of hierarchical surplus and hierarchical residual require (6) to hold.
(2) our element-wise operator evaluation relies on the fact that all vertices adjacent to this cell hold valid values. If some of them hold values that are not injected yet from a finer grid, the residual evaluation yields wrong results.
(3) our algorithm shall be allowed to refine and erase without restrictions. If (6) holds, the deletion of whole subtrees is allowed. Otherwise, valid values first have to be injected prior to a grid update.

Additional tree traversals reconstructing (6) on-demand require multiple unknown reads and writes. Notably, reconstruction schemes might run into a rippling effect [Sundar et al. 2008] where an update implies follow-up updates. All techniques introduced from hereon avoid additional data access and advocate for the minimisation of multiscale grid sweeps. This makes them future-proof regarding a widening memory gap, i.e. growing latency and shrinking bandwidth per core [Kowarschik et al. 2000].

They rely on two ingredients. On the one hand, we shift the additive algorithm's unknown updates by half an iteration, i.e. grid sweep. We run through the spacetree and determine unknown corrections to the current approximate solution. However, we do not feed them back into the solution immediately. Instead, we postpone the update and apply them in the beginning of the next solver cycle. On the other hand, we introduce two helper variables keeping track of updates w.r.t. the solution and, hence, also the injection. We preserve (6) due to an exchange of updates. We never compute the injection directly.

All ideas materialise in Algorithm 2 and realise the following invariant:

$$\begin{aligned} sc_\ell &\equiv 0 \qquad \forall \ell < \ell_{min}, \qquad \text{and} \\ sf_\ell &= sc_\ell = 0 \qquad \text{at startup.} \end{aligned}$$

Different to Algorithm 1 our realisation relies on a top-down tree traversal: We start at the coarsest level rather than on the bottom of the tree. Dynamic refinement thus remains simple. Whenever a spacetree cell is traversed that has to be refined, one

ALGORITHM 2: Additive multigrid with FAS that integrates into a tree traversal, i.e. a coarse-to-fine sweep through the grid hierarchies. A call to TDADD(ℓ_{min}) starts one multigrid cycle. sc and sf are helper variables introduced by pipelining that facilitate a single-touched policy. The transition from a correction scheme into FAS through the temporary helper variables $\hat{r}$ (hierarchical residual) and $\hat{u}$ (hierarchical surplus) is illustrated in Algorithm 4 in the appendix.

```
1:  function TDADD(ℓ)
2:     sc_ℓ ← sc_ℓ + P sc_{ℓ-1}          ▷ Add coarse grid correction to sc_ℓ which
3:                                       ▷ so far, holds update resulting from a Jacobi smoothing step.
4:     u_ℓ ← u_ℓ + sc_ℓ + sf_ℓ           ▷ Update u with update from
5:                                       ▷ previous line plus all updates done on finer grids.
6:     û ← u_ℓ − P u_{ℓ-1}                ▷ Determine new hierarchical surplus.
7:     if ℓ < ℓ_max then
8:        TDADD(ℓ + 1)
       end
                                         ▷ Go to next finer level.
9:     r_ℓ ← b_ℓ − H_ℓ u_ℓ                ▷ Determine residual and
10:    r̂_ℓ ← b_ℓ − H_ℓ û_ℓ                ▷ hierarchical residual.
11:    sc_ℓ ← ω_ℓ S(u_ℓ, b_ℓ)             ▷ Bookmark update due to a Jacobi
12:                                      ▷ smoothing step for next traversal. Usually
13:                                      ▷ uses r, otherwise there is no need to compute r.
14:    if ℓ > ℓ_min then
15:       b_{ℓ-1} ← R r̂_ℓ                ▷ Determine right-hand side
16:                                      ▷ for multigrid correction.
17:       sf_{ℓ-1} ← I (sf_ℓ + sc_ℓ)     ▷ Inform coarser grid about update
18:                                      ▷ due to smoothing that will happen in next traversal.
       end
19: end function
```

adds an arbitrary number of levels, initialises all hierarchical surpluses with zero and immediately descends into the new grid entities. Higher order prolongation can be constructed starting from this p-linear scheme.

Remark 5.2. The single sweep FAS facilitates arbitrary on-the-fly refinement. Such a dynamic refinement plays two roles. On the one hand, it allows the algorithm to resolve local solution characteristics. On the other hand, it yields a full multigrid (FMG)-like algorithm. We start from a coarse grid tackled by a series of additive V-cycles. Throughout these solves, we dynamically and locally add additional grid levels and thus unfold a coarse start grid into an adaptively refined accurate grid. Coarse solution approximations act as initial guesses for refined grids.

We note that our pipelined implementation requires us to store an additional two values sc and sf per vertex. They hold smoothing contributions (therefore the symbol s) from the coarser grids (symbol c) plus the current grid or smoothing contributions from the finer grids (symbol f). Besides these two values, also $\hat{r}$ and $\hat{u}$ have to be held. However, we may discard those in-between two iterations, and so they do not permanently increase the memory footprint.

While the present approach picks up ideas of pipelining—rather than multiple data consistency sweeps, we need only one amortised traversal per update; an achievement made possible due to additional helper variables (cmp. [Ghysels et al. 2013; Ghysels and Vanroose 2014; 2015])—the exchange of updates might induce stability problems, i.e. values on different levels that should hold the same values might diverge. We analysed this effect and studied the impact of a resync after a few iterations, but for none of the present experiments this resync has proven to be necessary.

5.1. ω choices

Algorithms 1 and 2 rely on relaxation parameters which have severe impact on the efficiency and stability of the resulting multigrid solver. Proper choices deliver several well-established multigrid flavours. While the semantics of the Jacobi relaxation ω_S in Algorithm 1 is well-understood and can be studied in terms of local Fourier analysis, Algorithm 1 also introduces ω_{cg}-scaling of the coarse grid contribution. Multiple valid choices for this parameter do exist with different properties [Bastian et al. 1998]. In Algorithm 2, we refrain from distinguishing ω_S and ω_{cg} in the formula but instead introduce a vertex-dependent relaxation ω_ℓ, i.e. each vertex may have its individual relaxation factor. As a vertex is unique due to its spatial position plus its level, this facilitates level-dependent ω choices.

Let $succ(v) \in \{0, \ldots, \ell_{max} - \ell_{min}\}$ be an integer variable per vertex v with

$$succ(v) = \begin{cases} 0 & \text{if } v \text{ is an unrefined vertex, or} \\ \min_i(succ(v_i)) + 1 & \text{for all children } v_i \text{ of } v \text{ otherwise.} \end{cases}$$

A child vertex of a parent vertex is a vertex with at least one adjacent cell whose parent in turn is adjacent to the parent vertex. This property deduces from the parent-child relation on the tree. Furthermore, we define the predicate $cPoint$ that holds for any vertex whose spatial position coincides with a vertex position on the next coarser levels. The predicate distinguishes c-points from f-points in the multigrid terminology. We obtain various smoother variants:

Relaxation parameter	Description
$\omega_\ell(v) = \begin{cases} \omega_S < 1 & \text{if } succ(v) = 0 \text{ and} \\ 0 & \text{otherwise.} \end{cases}$	Relaxed Jacobi on the dynamically adaptive grid as the *coarse relaxation parameter equals zero*.
$\omega_\ell(v) = \omega_S < 1$	*Undamped coarse grid correction*.
$\omega_\ell(v) = \begin{cases} \omega_S < 1 & \text{if } succ(v) \leq L \text{ and} \\ 0 & \text{otherwise} \end{cases}$	*Undamped L-grid* scheme on adaptive grids
$\omega_\ell(v) = (\omega_S)^{succ(v)+1} \quad \omega_S < 1$	Classic additive multigrid from [Bastian et al. 1998] where coarse grid updates contribute to the fine grid solution with an *exponential damping*.
$\omega_\ell(v) = (\omega_S)^{(1-1/n)\cdot(succ(v)+1)}$	*Transition* relaxation. n is the iteration counter.

Schemes that use the same ω on each and every level (undamped coarse grid correction) become unstable [Bastian et al. 1998] for setups with many levels. They tend to overshoot. If only one or two grids are used for the multigrid scheme ($L \in \{2, 3\}$), this overshooting is not that significant and solvers are more robust. However, we loose multigrid efficiency. Exponential damping is thus used by most codes. The coarser the grid the smaller its influence on the actual solution. This renders an exact coarse grid solve unnecessary. With the relaxation factors from above all the schemes apply straightforwardly to dynamically adaptive grids. Empirically, we observe that undamped schemes outperform their stable counterparts in the first few iterations. The overshooting is not dominant yet. We therefore propose a hybrid smoother choice that transitions from an undamped coarse grid correction into exponential damping.

5.2. Hierarchical basis and BPX-type solvers

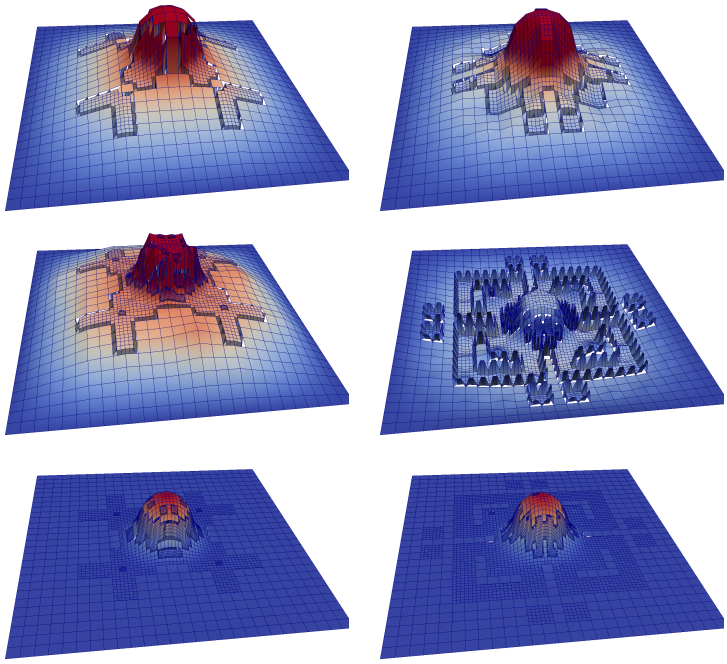

Fig. 5. Snapshot of the solution of $-\Delta u + 1000\ u = \chi$ with the transition scheme of the plain additive multigrid (left) and the hierarchical basis approach (right) after 5, 8 and 18 iterations. χ is a load of one within a circle around the domain's centre, i.e. a characteristic function which can be modelled via a Heaviside operator. For $-\phi \approx 4000$, the plain additive scheme left becomes unstable due to overshooting already visible here. The `hb`-scheme updates per level only additional fine grid points compared to the coarser grids and thus is less sensitive to overshooting. Both visualisations set hanging nodes to value zero. The rough components in the pictures thus are visualisation artefacts; no real high frequency contributions.

While additive multigrid with exponential damping or transition is robust for the Poisson equation, it runs into instabilities if we encounter a non-zero $\phi < 0$ term in (3); despite the fact that the problem remains well-posed positive-definite on all levels due to the additional minus sign in front of ϕ. The solver is sensitive to the reaction term. Robustness with respect to a reaction term however is mandatory prior to tackling any ill-definiteness.

We find the shift $\phi < 0$ make the additive multigrid overshoot on coarser levels, pollute the approximation and introduce a non-local oscillation in the follow-up iteration. The overshooting/oscillation typically grows per iteration if the diffusive operator is not dominating (Figure 5). A straightforward fix to this instability is the switch from a hierarchical generating system into a hierarchical basis [Griebel 1990; 1994]. It is identified by an `hb-` prefix from here on. Following [Bastian et al. 1998], such a switch results from a modification of the generic relaxation parameter into

$$\omega_\ell(v) \leftarrow \begin{cases} 0 & \text{if } cPoint(v) \quad \text{and} \\ \omega_\ell(v) & \text{otherwise.} \end{cases} . \tag{8}$$

We mask out c-points. Such a modification unfolds a variety of reasonable and unreasonable smoothing schemes due to the various choices of $\omega_\ell(v)$ on the right-hand side. Our numerical results detail this.

While (8) with its localised updates—vertices coinciding spatially are updated solely on the coarsest level—prevents the additive scheme from overshooting too significantly for $\phi < 0$, it comes at the price of a deteriorating convergence speed. It continues to assume a uniform smooth geometric multiscale behaviour of the solution, as any unknown update is determined by the update in the point plus the c-point updates of surrounding vertices. Increasing absolute values of ϕ in combination with non-smooth right-hand sides however decreases the smoothness of the solution along jumps of the latter. This becomes apparent immediately at hands of a gedankenexperiment with a Heavyside χ. A `hb`-solver locally overshoots where χ changes and the overall approximation starts to creep towards the correct solution due to local oscillations while non-local oscillations are eliminated.

One fix to this challenge adds an additional $-PI\omega_\ell S(u_\ell, b_\ell)$ term to all smoothing updates. This is known as BPX [Bastian et al. 1998; Bramble et al. 1990]. Reiterating through our data dependency analysis, we find that the BPX operator can not be implemented straightforwardly within Algorithm 2—even with the pipelining variables in place—as c-point impacts spread to their surrounding through the coarser grids while they are not altered themselves.

These considerations lead to a BPX FAS in Algorithm 3. The key idea is to keep sf and sc and to introduce another helper variable si holding the injected value of a smoother update without any c-point distinction. It is set as soon as we determine the smoother impact. This impact is discarded for c-points due to (8). Finally, the one-sweep realisation modifies the prolongation by adding an additional $-Psi$ term. (8) in combination with this term ensures the BPX inter-level correlation as we have $(id - PI) = 0$ on vertices for which $cPoint$ holds.

ALGORITHM 3: Single-sweep BPX variant realisation incorporating FAS. Invoked by TDBPX(ℓ_{min}). We do not rely on (8) here, i.e. ω is $cPoint$-agnostic, as we realise the case distinction within the multilevel code.

function TDBPX(ℓ)
  $sc_\ell \leftarrow sc_\ell + Psc_{\ell-1}$
  if *not* $cPoint(v)$ **then**
    $sc_\ell \leftarrow sc_\ell - Psi_{\ell-1}$ ▷ BPX-type modification of fine grid correction
  end
  $u_\ell \leftarrow u_\ell + sc_\ell + sf_\ell$
  $\hat{u} \leftarrow u_\ell - Pu_{\ell-1}$
  if $\ell < \ell_{max}$ **then**
    TDBPX($\ell + 1$)
  end
  $r_\ell \leftarrow b_\ell - H_\ell u_\ell$
  $\hat{r}_\ell \leftarrow b_\ell - H_\ell \hat{u}_\ell$
  if $cPoint(v)$ **then**
    $sc_\ell(v) \leftarrow 0$ ▷ Realise (8), i.e. cancel out update
  else
    $sc_\ell \leftarrow \omega_\ell S(u_\ell, b_\ell)$ ▷ Anticipate coarse correction
  end
  if $\ell > \ell_{min}$ **then**
    $si_{\ell-1} \leftarrow I\omega_\ell S(u_\ell, b_\ell)$ ▷ Memorise dropped fine grid update
    $b_{\ell-1} \leftarrow R\hat{r}_\ell$
    $sf_{\ell-1} \leftarrow I\,(sf_\ell + sc_\ell)$
  end
end function

We emphasise that (8) follows the same pipelining idea we introduced for the additive scheme and at the same time renders the storage of a fine grid correction sf unnecessary. We could add any fine smoothing impact directly onto sc and at the same time skip the injection of $sf_\ell + sc_\ell$. Such a BPX realisation uses the same data layout as the additive multigrid. No sf is to be held, but we need an additional si. This preserves the number of variables. The reason for this possibility results from the fact that the coarsest vertex in Ω_h holds the valid nodal representation of the solution in Algorithm 3. All finer vertices at the same location are copies. We preserve sf in the presented code to emphasise the closeness to the additive scheme. While this wastes one entry per vertex, it might make sense to preserve the fine grid injection and thus to allow BPX's fine grid update to change $cPoints$ as well: in applications with non-trivial boundary conditions, those sometimes are simpler to evaluate in a nodal setting rather than a hierarchical basis. The injection then automatically reconstructs the data consistency on all levels.

5.3. Feature-based dynamic adaptivity

All algorithmic ingredients introduced are well-suited for any arbitrary adaptivity. Throughout the top down steps, we may add any number of vertices as long as we initialise their hierarchical surplus with 0 and prolong the solution p-linearly. They then seamlessly integrate into the solver's workflow. For faster convergence, higher order interpolation might be advantageous. Discarding vertices is permitted throughout the backtracking, i.e. the steps up in the grid hierarchy. The FAS ensures that all solution information is already available on the coarsened mesh. Multilevel meta information such as $cPoint$ or $succ(v)$ can be computed on-the-fly throughout the tree traversal's backtracking. It then automatically adopts to updated refinement patterns.

In the present paper, we stick to simple feature-based refinement and specify both regular grids and adaptive grids through a maximal and minimal mesh size h_{max} and h_{min}. We start from a grid satisfying h_{max} and, in parallel to the smoothing steps, measure the value $s = \max_{d\in\{1,\ldots,p\}} |\Delta_d u|$ per vertex on each grid level. Per step, we refine the 10% of the vertices with the highest s value, while we erase the 2% vertices with the smallest s value. These values are shots from the hip but empirically show reasonable grid refinement structures. They yield a grid that adopts itself to solution characteristics. We realise feature-based adaptivity. More sophisticated schemes with proper error estimators are out of scope.

To avoid global sorting, we split up the whole span of s values into 20 subranges and bin vertices into these ranges. All vertices fitting into a fixed number of bins holding the largest s values are refined. This fixed number is selected such that the 10% goal is met as close as possible. Erasing works analogously with the bins with the smallest s values. Refining and erasing are vetoed in two cases: if maximal or minimal mesh constraints would be violated; or if residual divided by diagonal element exceeds 10^{-2}. In the latter case, the vertex is still subject to major updates, i.e. has not 'converged', and we postpone a refinement or coarsening.

The interplay of the feature-based refinement with the creation of a FMG cycle is detailed in Remark 5.2. We note that our criterion yields different grid refinement patterns for different solvers as we integrate refinement into the solve (Figure 5). This advocates for better criteria and renders the present experiments feasibility studies.

6. SOLVER PROPERTIES

In the following, we validate fundamental properties of the algorithm. We prove its correctness. Let the iterates of an unknown x be $x^{(n)}, x^{(n+1)}, \ldots$ As sc is updated twice per iteration we distinguish $sc^{(n)}$, $sc^{(n+0.5)}$ and $sc^{(n+1)}$.

LEMMA 6.1. *Whenever we evaluate* $r_\ell^{(n+1)} = b_\ell - H_\ell u_\ell^{(n)}$*, we have*

$$\forall n: \qquad u_{\ell-1}^{(n)} = I u_\ell^{(n)} \qquad \forall \ell_{min} < \ell \leq \ell_{max}.$$

PROOF.
At construction, $u_\ell^{(0)} = I u_{\ell+1}^{(0)}$. The proof then relies on a simple induction over n:

$$\begin{aligned}
u_{\ell-1}^{(n+1)} &= u_{\ell-1}^{(n)} + sf_{\ell-1}^{(n+1)} + sc_{\ell-1}^{(n+1)} \\
&= I u_\ell^{(n)} + I\left(sf_\ell^{(n+1)} + sc_\ell^{(n+1/2)}\right) + sc_{\ell-1}^{(n+1)} \\
&= I u_\ell^{(n)} + I s f_\ell^{(n+1)} + I\left(sc_\ell^{(n+1)} - P sc_{\ell-1}^{(n+1)}\right) + sc_{\ell-1}^{(n+1)} \\
&= I u_\ell^{(n)} + I s f_\ell^{(n+1)} + I sc_\ell^{(n+1)} + (id - IP) sc_{\ell-1}^{(n+1)} \\
&= I u_\ell^{(n)}
\end{aligned}$$

where we apply the induction hypothesis on $u_\ell^{(n-1)}$, the algorithm's update operations on the remaining operators, and exploit $IP = id$ for c-points. Dynamic adaptivity has to preserve the construction constraint. □

The smoother here is a black-box and we do not make any assumption about the correctness of the solved equation systems. Our notation of I, P and R further is generic, i.e. I^k indicates that I is applied multiplicatively k times in a row with an I fitting to the preimage.

The lemma implies that transitions between grid resolutions require no special treatment: As each vertex on each grid level holds a valid representation of the solution, we can apply the same stencils irrespective whether they overlap a refined region or are fine grid stencils. Only hanging nodes have to be interpolated p-linearly from the coarser grid.

LEMMA 6.2. *The operators from Algorithm 2 realise a FAS.*

PROOF. Each level's smoother tackles a correction equation of the form

$$H_\ell\left(u_\ell + I u_{\ell+1}\right) = R r_{\ell+1} + H_\ell I u_{\ell+1}.$$

The left-hand side has been studied before. So we follow [Griebel 1994] and write

$$\begin{aligned}
R\hat{r}_{\ell+1} &= R\left(b_{\ell+1} - H_{\ell+1}\hat{u}_{\ell+1}\right) \\
&= R\left(b_{\ell+1} - H_{\ell+1}(u_{\ell+1} - P u_\ell)\right) \\
&= R r_{\ell+1} + R H_{\ell+1} P u_\ell = R r_{\ell+1} + R H_{\ell+1} P I u_{\ell+1} \quad \text{(cmp. Lemma 1)} \\
&= R r_{\ell+1} + H_\ell I u_{\ell+1}.
\end{aligned}$$

□

The proof holds for space-independent Helmholtz shifts ϕ and Dirichlet and Neumann boundary conditions. As mentioned before, absorbing boundary layers with varying complex scaling or jumping material coefficients violate the equivalence of a Galerkin multigrid operator and plain rediscretisation. We have to assume that these are the regions many adaptivity criteria refine towards to. For these, the equations above comprise an additional error term if we do not apply operator-dependent grid transfer operators. The multigrid equations are perturbed. Based on empirical evidence, we assume this perturbation to be small. However, we have to expect a small deterioration of the multigrid performance. Without pollution, the prolongation of

$$u_\ell^{(n+1)} - u_\ell^{(n)} = \omega_{\ell-1} S(u_\ell^{(n)}, b_\ell) = cs_\ell^{(n+1)}$$

to level $\ell+1$ equals the correction term in multigrid.

Remark 6.3. The lemma extends naturally to Algorithm 3, i.e. the BPX variant.

THEOREM 6.4. *The additive top-down FAS from Algorithm 2 realises an additive multigrid algorithm.*

PROOF. We compare the algorithm to additive blueprint in Algorithm 1 and study one iteration of the scheme.

— We first study the one-grid problem with $\ell_{min} = \ell_{max}$. We further focus on $u_\ell^{(n+1)}$, i.e. work backwards from the update of this unknown.

$$\begin{aligned} u_\ell^{(n+1)} &= u_\ell^{(n)} + Psc_{\ell-1}^{(n+1)} + sc_\ell^{(n+1)} + sf_\ell^{(n+1)} \\ &= u_\ell^{(n)} + sc_\ell^{(n+1)} + sf_\ell^{(n+1)} \qquad \text{as } sc_{\ell-1} \equiv 0 \\ &= u_\ell^{(n)} + \omega_\ell S(u_\ell^{(n)}, b_\ell) + sf_\ell^{(n+1)} \end{aligned}$$

sf_ℓ is never modified which closes this step.

— We next switch to a two-grid problem with $\ell \equiv \ell_{max}$ and $\ell - 1 \equiv \ell_{min}$.

$$\begin{aligned} u_\ell^{(n+1)} &= u_\ell^{(n)} + Psc_{\ell-1}^{(n+1)} + sc_\ell^{(n+1)} + sf_\ell^{(n+1)} \\ &= u_\ell^{(n)} + Psc_{\ell-1}^{(n+1)} + sc_\ell^{(n+1)} \qquad \text{induction} \\ &= u_\ell^{(n)} + Psc_{\ell-1}^{(n+1)} + \omega_\ell S(u_\ell^{(n)}, b_\ell) \\ &= u_\ell^{(n)} + P\left(\omega_{\ell-1} S(u_\ell^{(n)}, b_\ell)\right) + \omega_\ell S(u_\ell^{(n)}, b_\ell) \end{aligned}$$

The proof follows from induction on the grid levels if we choose $\omega_\ell = \omega_S$ on the finest level and $\omega_\ell = \omega_S \cdot \omega_{cg}^l$ with l being the difference of the current level to the finest level. The latter is an attribute that can be computed on-the-fly throughout the bottom-up steps of the algorithm within the spacetree. □

Remark 6.5. We assume that a theorem for the BPX solver from Algorithm 3 is proven analogously.

7. RESULTS

Our results split into five parts. First, we study the convergence behaviour of the additive multigrid variants for a simple Poisson equation. This validates the algorithmic building blocks at hands of a well-posed setup and yields insight into the convergence efficiency. Second, we switch to Helmholtz problems with $\phi < 0$ in (3). This reveals the shortcomings of the additive scheme compared to a hierarchical basis approach that arises naturally from our chosen data structures. Third, we study Helmholtz problems with $\phi > 0$, i.e. the difficult case of ill-conditioned problems. This quantifies the efficiency and robustness of the proposed solution with complex grid rotation. Fourth, we apply our toolset on the motivating scattering example. This validates that the approach is well-suited to tackle varying material parameters ϕ. Finally, we study the efficiency of the proposed solver regarding hardware characteristics. For the present perfectly parallel setup, this notably has to focus on vectorisation and memory access efficiency. The latter case studies are conducted either on a local workstation with Sandy Bridge-EP Xeon E5-2650 processors running at 2.0 GHz or on a Xeon Phi 5110P accelerator running at 1,053 GHz. The latter is a pioneer of future manycore architectures delivering performance to a significant extent through vectorisation. At the same time, it is well-known to be sensitive to proper memory layout and access [Reinders and

Jeffers 2015]. Statements on this machine thus facilitate an extrapolation to upcoming hardware generations. Our codes were translated with the Intel compiler 15.0.1, while we used the Likwid tools [Treibig et al. 2010] to obtain statements on hardware counters.

Global quantities are specified over all fine grid vertices, i.e. a subset of $\mathbb{V}$; either in the maximum norm, the standard Euclidean norm or

$$|x|_h^2 = \sum_i |h_i|^d |x_i|^2,$$

where each entry of the input vector is scaled by the volume of one spacetree cell of the corresponding mesh level. While the $_h$-norm is an Euclidean inner product norm (cmp. (1.3.6) from [Trottenberg et al. 2001]) anticipating nonuniform mesh elements and thus allows us to compare adaptive grids with each other, we emphasise that it is not an exact equivalent of the continuous L2 norm as small slices of the domain along adaptivity boundaries are integrated several times due to our hierarchical ansatz space. For regular grids, it is exact. The norm allows us to compare grids of different resolution [Trottenberg et al. 2001] or adaptivity patterns with each other. We thus have to expect minor peaks in the residual whenever the grid is dynamically refined.

$$\chi(x) = d\pi^2 \cdot \prod_{i=0,\ldots,d} \sin(\pi x_i) \qquad \text{and} \tag{9}$$

$$\chi(x) = \begin{cases} 1, & \text{if } \sum_{j=1}^{p} (x_j - \frac{1}{2})^2 < 0.1^2, \\ 0, & \text{elsewhere.} \end{cases} \tag{10}$$

act as artificial benchmark problems before we switch to realistic setups in Section 7.4. Homogeneous Dirichlet boundary conditions are supplemented for the benchmarks.

7.1. Poisson problems

We start from (9) and $\phi \equiv 0$. The complex rotation is $\theta = 0$. An analytical solution to this problem has no imaginary parts, and we have validated for all choices of $\omega_\ell(v)$ the convergence towards the analytic solution. A convergence study on uniform grids exhibits a convergence speed that is almost mesh-independent for the different multigrid variants (Figure 6). We observe that the speed slightly decreases for decreasing mesh width if we use exponential or transition coarse grid damping. While these two schemes are the robust multiscale variants, they downscale the impact of coarse grid corrections per additional coarse grid level. We can not expect perfect mesh-independent convergence. In general, the transition scheme slightly outperforms the exponential coarse grid damping after a few additive cycles. Undamped additive multigrid converges only for the coarser mesh widths—it would still work if we reduced ω with each additional grid level; which decreases the convergence speed—while the three grid and Jacobi solver exhibit the well-known mesh-dependent convergence behaviour.

The aforementioned convergence statements reveal to be too pessimistic if we switch to an adaptive, FMG-type setting. We then observe that we obtain two orders of accuracy at the cost of around three fine grid cycles (Figure 7). This holds for both the exponential coarse grid damping (not shown) and the transition scheme while the latter performs slightly better again. We are close to multiplicative multigrid efficiency.

We finally observe that optimal, i.e. mesh-independent, convergence is obtained for different dimensions if $\omega = 0.8$. While larger values make the solver diverge, stronger damping such as $\omega = 0.6$, e.g., does not deliver comparable performance robustly

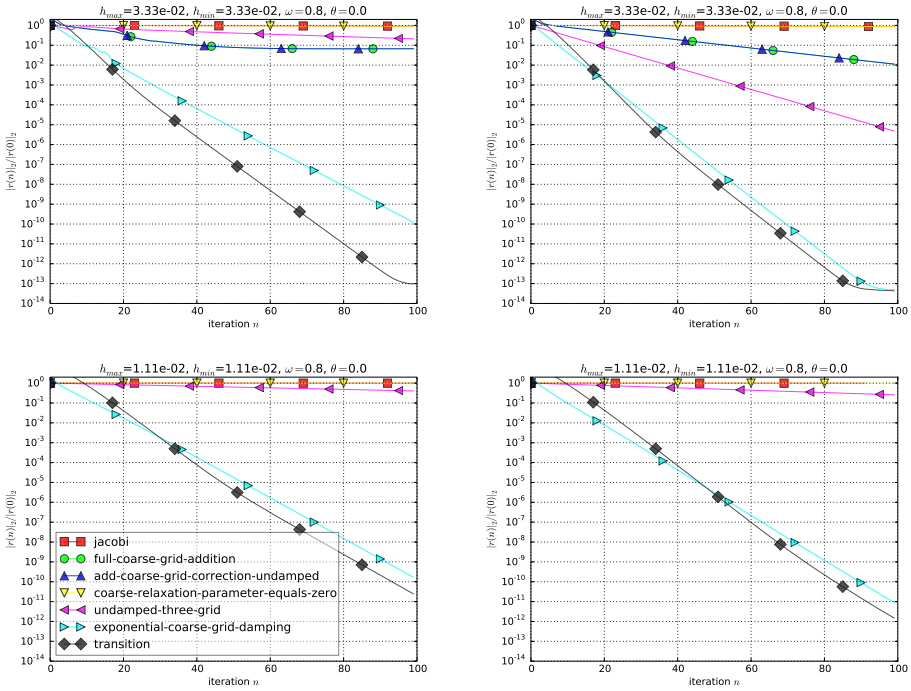

Fig. 6. Residual development on regular grids (left: $p = 2$; right: $p = 3$) for the additive multigrid and the right-hand side (9).

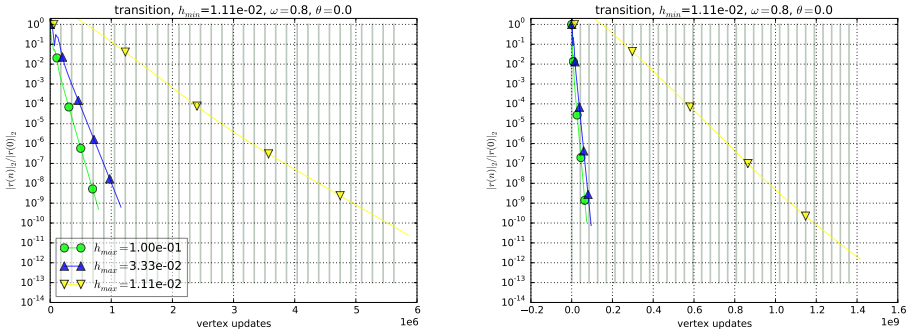

Fig. 7. Adaptive grid with additive multigrid that is successively refined from a prescribed maximum mesh size h_{max} to the minimum mesh size h_{min} for $p = 2$ (left) and $p = 3$ (right) where it pays off. Each thick vertical line denotes the cost of three additive cycles on a regular grid with h_{min}. Transition is used for $\omega_\ell(v)$. The right-hand side is (9).

though some mesh choices benefit from reduced factors (Figure 8). Again, exponential damping and transition are the only schemes that are stable for all $p \in \{2, 3, 4\}$ and all mesh size configurations. Transition usually is faster than exponential damping. However, the convergence speed of both approaches continues to deteriorate with increasing p. Reasons for this might be found in the poor smoother and the rather aggressive coarsening by a factor of 3^p.

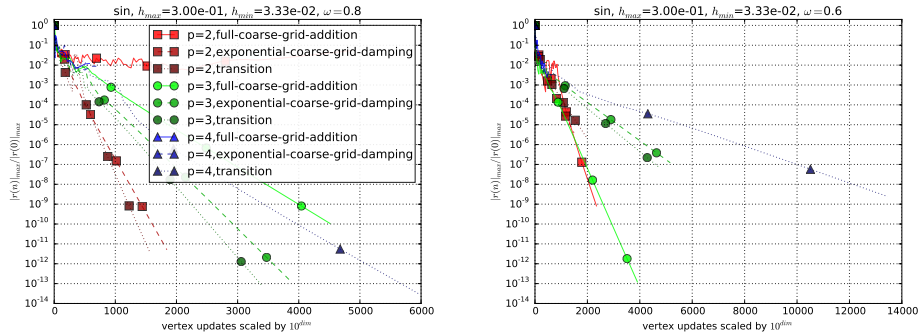

Fig. 8. Comparisons of the solver efficiency for different dimensions for one adaptive mesh configuration choice. Note the dimension-dependent scaling of the x-axis.

Table I. Cost to reduce the initial residual by 10^{-6}. The first entry is the number of grid sweeps required by the adaptive solvers, the second entry normalises the required unknown updates: It shows how many regular grid sweeps yield the same cost. `ucg` denotes undamped coarse grid correction, `exp` exponential coarse grid damping and `t` transition. $p = 2$. The upper part shows results for (9), the lower for (10). $\perp$ denotes divergence, * setups where the adaptivity criterion had not created stationary grid setups yet.

		additive multigrid			hb		bpx		
ϕ	h_{min}	ucg	exp	t	ucg	exp	ucg	exp	t
-400	0.05	32/27.6	38/32.8	35/30.2	183/157.8	197/169.9	31/26.7	35/30.2	32/27.6
	0.005	$\perp$	67/7.7	70/10.6	370/44.5	517/55.9	49/8.9	74/11.3	69/10.5
	0.0005	$\perp$	95/8.0	97/7.2	$\perp$	$\perp$	65/*	93/6.3	95/6.7
-4000	0.05	32/27.6	38/32.8	35/30.2	183/157.8	197/169.9	31/26.7	35/30.2	32/27.6
	0.005	$\perp$	67/7.7	70/10.6	370/44.5	517/55.9	49/8.9	74/11.3	69/10.5
	0.0005	$\perp$	95/8.0	97/7.2	$\perp$	$\perp$	65/*	93/6.3	95/6.7
-200	0.05	46/39.7	29/25.0	28/24.1	254/219.0	332/286.3	29/25.0	29/25.0	28/24.1
	0.005	$\perp$	64/3.7	60/3.4	544/4.3	≥ 1000	61/3.5	63/3.4	62/0.4
	0.0005	$\perp$	196/*	$\perp$	$\perp$	$\perp$	107/*	71/0.9	$\perp$
-1000	0.05	$\perp$	$\perp$	$\perp$	$\perp$	651/561.3	18/15.5	18/15.5	17/14.7
	0.005	$\perp$	46/2.6	61/3.5	$\perp$	≥ 1000	50/2.3	73/0.4	61/2.9
	0.0005	$\perp$	$\perp$	115/*	$\perp$	≥ 1000	58/0.1	$\perp$	101/0.2
-4000	0.05	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	26/22.4	33/28.5	31/26.7
	0.005	$\perp$	$\perp$	$\perp$	$\perp$	≥ 1000	36/1.3	46/1.7	76/2.7
	0.0005	$\perp$	148/*	181/*	$\perp$	≥ 1000	97/*	188/*	243/*

We summarise that our approach works for $p \geq 2$, but remains not practical for $p \geq 5$ due to the curse of dimensions. To the best of our knowledge, even results for a numerical solution with $p \geq 3$ in our application area are rare. The aforementioned [Baertschy and Li 2001; Vanroose et al. 2005; Horner et al. 2007; bin Zubair et al. 2012; Cools et al. 2014a] for example all restrict to $p \in \{2, 3\}$, i.e. two-dimensional or three-dimensional grids. To the best of our knowledge, very few spacetree codes offer four-dimensional or even higher-dimensional dynamically adaptive grids.

7.2. Case $\phi < 0$: positive definite Helmholtz problems

Negative ϕ in (3) with the right-hand side from (9) have a negligible impact on the convergence behaviour of the solvers (Table I; upper part). Here, additive multigrid and BPX yield comparable results. BPX seems to become superior for sufficiently fine grids. The hierarchical basis approach is slower. For BPX, undamped coarse grid damping is the method of choice. As we stop the study for rather big residuals being in the order of 10^{-6} relative to the start residual, the multigrid's transition scheme has not yet

overtaken the exponential coarse grid damping. We observe for all setups that finer mesh resolutions require more sweeps. The grid has to unfold completely due to these sweeps. The cost (in terms of unknown updates) normalised by the cost per sweep on a regular grid of the finest mesh size however decreases with additional levels. Inaccuracies occur for BPX that stopped right after the refinement criterion had inserted an additional grid level. It thus does not make sense to compare the number of updates to a regular grid—the new level that just had been inserted makes the adaptive scheme seem to be too good. Longer simulation runs/a lower termination threshold would put these values into perspective.

The convergence characteristics change for (10) acting as right-hand side (Table I; lower part). Additive multigrid starts to diverge for coarser mesh sizes already as $\phi < 0$ becomes smaller. For fine meshes, it always diverges. The ill-behaviour stems from the fact that our coarse grid updates mimic a long-range diffusive behaviour of the solution. The smaller $\phi < 0$ the less significant this diffusion component in (3). Instead, we face steep gradients at the transition of the right-hand. They can not be resolved on coarse grids. Even worse, any coarse grid change pollutes a fine grid approximation due to unnatural diffusion introduced by our p-linear P. It thus excites oscillations around the χ transition.

The hierarchical basis is more robust w.r.t. these non-diffusive oscillations if we use exponential damping. However, its convergence speed deteriorates. Exponential damping in combination with the `hb`-filtering of c-points on the fine grids yields a scheme where unknowns within the computational domain that are induced by coarse levels are not updated significantly anymore once finer grids are introduced; or once restricted residuals average out on coarse levels. Any combination of `hb` without full coarse grid addition of the correction thus makes only limited sense and is not followed-up further. Results for `hb` with transition are not even shown. `hb` seems to be a problematic solver variant here. It is due to the additive framework and might be completely different for multiplicative settings.

Our BPX-type variant finally yields the best results. BPX starts to reduce the cost per accuracy for shrinking ϕs unless its hits convergence (at a cost of around 0.1 regular fine grid sweeps). Hereby, an undamped coarse grid addition is superior to the other variants; as it materialises in the number of sweeps. As the PDE deteriorates to an explicit equation $-\phi\psi = \chi$ with a relatively small diffusive addendum on the left-hand side, fine grid unknowns introduced by coarse grid levels are updated almost to the right solution immediately due to the dominance of the diagonal in the system matrix. The residual for other unknowns is (almost) correct on the finest grid resolution, too. Where the multigrid and hierarchical basis update the latter points plus add a prolonged correction from the c-points—the latter update component over-relaxes the unknowns—BPX explicitly removes the coarse grid contribution, as the coarse grid contribution equals exactly the fine grid update.

7.3. Case $\phi > 0$: indefinite Helmholtz problems on complex rotated grids

Depending on the grid resolution and topology, (3) can become indefinite or yield a non-trivial null space for $\phi > 0$. Outgoing waves mimicked by absorbing boundary layers add natural damping to most of the eigenmodes and therefore make the discretised operator better conditioned. In the present section, we stick to homogeneous Dirichlet boundary conditions only, i.e. study a worst-case scenario regarding the numerical stability. In turn, we isolate the impact of complex grid rotation from any other damping induced by outgoing wave boundary conditions in typical applications.

In the following experiments, the mesh width is scaled according to $h \to he^{i\theta}$ in each dimension ($0 \leq \theta \leq \frac{\pi}{4}$). Complex rotation has a positive effect on the convergence

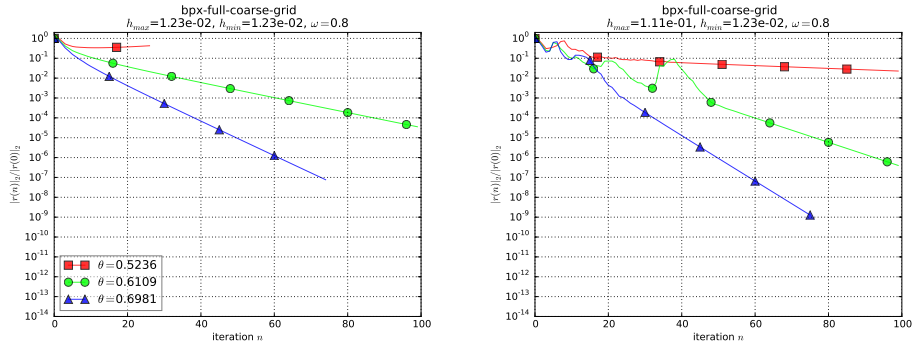

Fig. 9. Residual development for the BPX-variant in Algorithm 3 with undamped coarse grid correction for $\phi = 45^2$. Different values of the complex rotation angle $\theta = 30°, 35°, 40°$ were tested on a regular grid (left) and an adaptive grid (right).

behaviour of the multigrid solver (cmp. Figure 9 with $\phi = 45^2$), while $\theta = 0°$ makes the solver diverge. $\omega = 0.8$ is used throughout all experiments. If we rotate the constant mesh width over $\theta > 30°$, then the solver gets into the regime of convergence. We observe the same behaviour on an adaptive grid that starts on a coarse regular grid with $h_{min} = \frac{1}{9}$. During a refinement step the residual reduction might temporarily increase. Once the grid settles into a steady state, an asymptotic convergence rate is reached. It is better than the corresponding convergence rate on a regular grid with the same minimal mesh size. This is due to a reduced set of eigenvalues.

A typical stress test for indefinite Helmholtz solvers is the robustness as a function of increasing Helmholtz shift $\phi = k^2$. k is the *wave number*. For one-dimensional problems a common restriction on the mesh size h is given by the *ten-points-per-wave-length* rule, which translates into $kh < 0.625$. In the following two-dimensional experiments we keep $kh = \frac{5}{9} = 0.5555\ldots$ and test for different values of k on regular grids. Note that from a physical point of view a more stringent constraint on h is required in higher dimensions, such as $k^3h^2 = \mathcal{O}(1)$, in order to avoid pollution of the solution [Bayliss et al. 1985; Ihlenburg and Babuska 1995].

In Figure 10 the convergence behaviour of different solver variants is compared, similar to the experiments in Figure 6 for the Poisson equation. The mesh width is now complex rotated over $\theta = 35° \approx 0.6109$ in order to avoid divergence due to a non-trivial null space. The top left panel ($k = 5$) shows a nice reduction rate. For increasing wave numbers ($k = 15, 45, 135$), the term $\phi = k^2 > 0$ starts to dominate the PDE and the solvers suffer from the overshooting effects discussed in the previous section with $\phi < 0$. As expected, only the BPX-variants seem to cope with the highest wave number $k = 135$ in the bottom right panel.

The BPX-variants are further tested on adaptive grids in Figure 11 for values $k = 45$ and 135. We let the corresponding mesh sizes now determine the finest possible resolution in the grid, and start each solve on a coarse regular grid with $h_{max} = \frac{1}{9}$. The result is an FMG-type solver that adaptively refines. The same as for the Poisson experiments in Figure 7, the horizontal axis indicates the number vertex updates versus the residual normalised residual norm on the vertical axis. Again, there is a significant benefit from coarser regions in the grid. Adaptivity is a particularly useful functionality for the highly heterogeneous Helmholtz equations with space-dependent function $\phi = \phi(\mathrm{x})$, that arise in the quantum mechanical problems described in Section 2. A typical L-shaped refinement [bin Zubair et al. 2010] is desirable for a good representation of extremely localised waves close to the domain boundary (cmp. Figure 1 right).

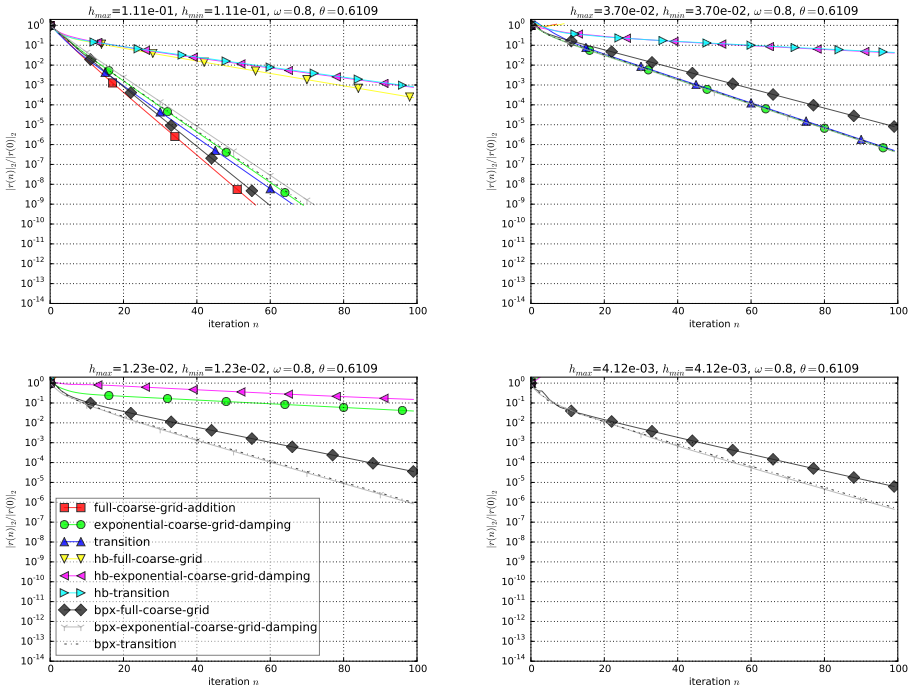

Fig. 10. Residual development for different values of the Helmholtz shift: $\phi = 5^2$ (top left), 15^2 (top right), 45^2 (bottom left) and 135^2 (bottom right). Complex rotation was set to $\theta = 35°$. The `hb-` prefix marks hierarchical basis solvers, the `bpx-` prefix a BPX approach.

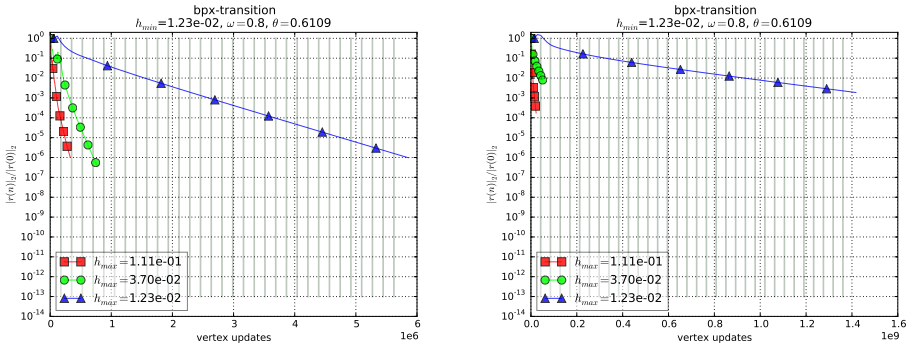

Fig. 11. Adaptive grid with BPX additive multigrid that is successively refined from a prescribed maximum mesh size h_{max} to the minimum mesh size h_{min} for $p = 2$ (left) and $p = 3$ (right). Helmholtz variant of the experiment in Figure 7, for $\phi = 45^2$ and complex rotation $\theta = 35°$.

Nonetheless, there remain severe numerical stability issues to handle these evanescent waves. We refer to the concluding Section 8 for an outlook on strategies, as this lies beyond the scope of the current paper.

7.4. $p = 2$ application scenario and grid adaptivity structure

With characteristics of the solver behaviour at hand, we finally study a realistic channel matrix for $p = 2$ idealised from the dynamics of Hydrogen or Deuterium [bin

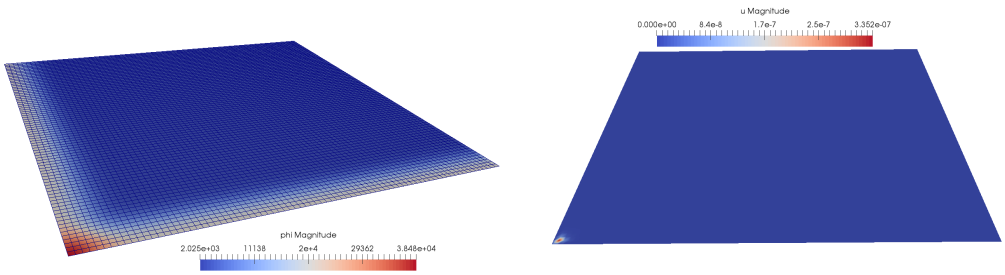

Fig. 12. The distribution of ϕ in the computational domain (left). It is invariant of the complex rotation. Complex rotation however does make a difference to the solution (right, $\theta = 45°$).

Zubair et al. 2012]. This leads in the channels' frequency domain to two-dimensional Helmholtz problems

$$\begin{aligned} \chi(x,y) &= e^{\left(-(125x)^2-(125y)^2\right)} \quad \text{and} \\ \phi(x,y) &= 45^2 + 135^2 \cdot \left(e^{-(15x)^2} + e^{-(15y)^2}\right), \end{aligned} \tag{11}$$

where the x-axis and the y-axis represent the distance from the centre of mass. They consequently carry homogeneous Dirichlet values: The probability for an electron to coincide with a nucleus equals zero. The remaining two faces top and right are open boundaries. They are consequently supplemented with homogeneous Dirichlet values as well, but we rotate the cells close to these faces by $30°$ in the complex domain to eliminate wave reflections. Close is $\frac{1}{3}$rd of the domain. The remaining cells in the domain are complex rotated by an angle $\theta \geq 0$, independently of this absorbing layer. As discussed in the previous section, only for $\theta = 0$ the original Helmholtz problem is solved. Both an inhomogeneous right-hand side as well as an inhomogeneous 'material' function $\phi > 0$ are active inside the domain. The actual system parameters are channel dependent and are determined by the potential fields of all present particles.

For this setup $h = 1/81$ is a physically reasonable choice for our purposes of studying the solution. The experiment is then sufficiently small to be solved by a direct solver. However, we solve it with pure Jacobi (cmp. Figure 1 right) which does not require us to change any code. For this, we use two-phase relaxation [Ernst and Gander 2011], i.e. two different relaxation parameters ω_1 and ω_2 alternatingly, which follows [Hadley 2005][1] with

$$\omega_1 = 0.01 \cdot \left(\sqrt{3} - 1\right) \quad \text{and} \quad \omega_2 = -\overline{\omega_1}.$$

One of the two relaxation parameters has a negative real part and both carry an imaginary component. Obviously, this approach becomes unfeasible once p increases or finer grid resolutions are required. For the additive multigrid (transition scheme) and the BPX (undamped coarse grid correction), we use $\omega = 0.4$ which avoids oscillations. We also use $\omega = 0.4$ for all Jacobi smoothers that apply complex rotations.

Though Jacobi converges for $\theta = 0$ and reduces the residual after every two grid sweeps, the convergence speed is unacceptably low even for this simple setup. How-

[1][Hadley 2005] uses a relaxation α that is relative to a finite difference discretisation of the Laplacian. In our case, we consequently have $\omega = \alpha(1 + \frac{h\phi}{4})$ with $\alpha = \sqrt{3} - 1$.

Table II. Reduction of residual of (11) in max norm after 50 iterations. The regular grid uses $1/81$, the adaptive one starts from $h_{max} = 0.1$ and does permit the criterion to refine until $h_{min} = 0.001$ is just underrun. $\perp$ denotes divergence. Figures in brackets show, if appropriate, the number of vertices used after 50 iterations.

	regular			adaptive		
θ	Jacobi	transition	BPX	Jacobi	transition	BPX
0°	$4.78 \cdot 10^{-1}$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
18°	$\perp$	$\perp$	$4.44 \cdot 10^{-2}$	$\perp$	$\perp$	$\perp$
25°	$2.07 \cdot 10^{-1}$	$2.27 \cdot 10^{-2}$	$2.62 \cdot 10^{-3}$	$9.82 \cdot 10^{-2}$ (1,536)	$5.30 \cdot 10^{-2}$ (1,448)	$5.72 \cdot 10^{-3}$ (1,500)
35°	$8.46 \cdot 10^{-5}$	$2.00 \cdot 10^{-4}$	$8.35 \cdot 10^{-4}$	$9.49 \cdot 10^{-2}$ (1,544)	$3.97 \cdot 10^{-2}$ (1,324)	$1.61 \cdot 10^{-3}$ (1,480)
45°	$6.53 \cdot 10^{-7}$	$6.67 \cdot 10^{-5}$	$2.60 \cdot 10^{-4}$	$9.14 \cdot 10^{-2}$ (1,524)	$3.09 \cdot 10^{-2}$ (1,308)	$1.16 \cdot 10^{-3}$ (1,460)

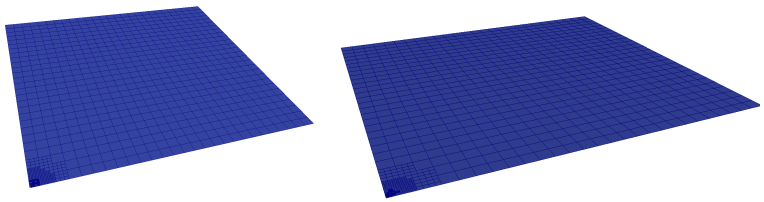

Fig. 13. Left: Dynamically adaptive grid with $h_{max} = 1/3$ and $h_{min} = 0.001$ for a pure Jacobi solver at the moment of convergence without any complex rotation. Right: Dynamically adaptive grid of BPX for the same setup with a complex rotation of $\theta = 45°$.

ever, the additive multigrid and BPX are not stable for $\theta = 0$ and thus can not be applied. BPX becomes stable for $\theta \geq 18°$, while the additive transition scheme requires $\theta \geq 25°$ (Table II). With increasing θ, the convergence speed of all solvers improves. This improvement is rendered problematic as the quality of the preconditioner suffers. While the latter effect is not directly studied here, it is indirectly illustrated by the Jacobi eventually outperforming the multigrid schemes. For large θ, all wave behaviour is damped out and we basically resolve one peak around the coordinate system's origin (cmp. Figure 12 right). The local solution characteristics render multilevel solvers inappropriate.

If we start from a grid of $h_{max} = 0.1$ and allow the dynamic refinement criterion to refine any cell coarser than $h_{min} = 0.001$, all solution processes automatically yield an adaptive grid (Figure 13). For all rotation choices, we end up with 1,308–1,544 vertices; a significant saving compared to a regular grid with 6,400 vertices. The purely feature-based criterion here has a two-fold role: In accordance with previous results, it rewrites the solve into an FMG-type cycle. At the same time, it allows the code to make the solution resolve the physical problem characteristics economically. While these solution characteristics depend sensitively on the choice of rotation, we observe that the adaptivity structure is almost rotation-invariant. This is an effect that deserves further studies but obviously results from the fact that the gradient around the Gaussian stimulus χ exceeds by magnitudes any characteristic of the induced wave pattern. Different to the regular grid results, we furthermore observe that BPX remains superior to the other approaches for all θ, while all variants outperform their regular grid counterparts in terms of cost: each adaptive iteration is at most 1/4th of the cost of a regular grid sweep.

Table III. Performance counters on the Sandy Bridge for a dynamically adaptive FMG-type solve with $h_{min} = 0.001$ ($p = 2$), $h_{min} = 0.005$ ($p = 3$), $h_{min} = 0.01$ ($p = 4$). The left value is obtained without vectorisation, the right results from an executable translated with simd facilities. L2 misses are relative measures (rates) compared to total number of accesses.

metric	p=2	p=3	p=4
Runtime	1.16e+02/1.13e+02	2.12e+02/2.11e+02	2.07e+03/2.31e+03
Instructions	5.53e+11/5.53e+11	1.11e+12/1.09e+12	1.19e+13/1.11e+13
MFlops/s	7.27e+02/8.89e+02	1.13e+03/1.40e+03	1.66e+03/1.98e+03
L2 requests	1.77e-02/1.74e-02	1.48e-02/1.23e-02	9.55e-03/1.04e-02
L2 misses	2.75e-03/2.77e-03	3.49e-03/3.00e-03	6.66e-04/8.29e-04
Used bandwidth (MB/s)	2.53e+02/2.52e+02	1.35e+02/1.37e+02	7.80e+01/7.14e+01
Transferred memory (GB)	2.90e+01	2.86e+01	1.65e+02

7.5. Hardware efficiency

We finally study the algorithm's hardware characteristic and compare the memory throughput to the Stream benchmark [McCalpin 1995] ran on a single core of the Sandy Bridge. An excellent cache usage mirrors results from [Weinzierl 2009; Weinzierl and Mehl 2011]. It results from the combination of strict element-wise data access, stack-based data management and depth-first spacetree traversal along a space-filling curve (Table III). Element-wise formulation and depth-first fit, i.e. localised data access even for the grid transfer operations, are characteristics of the present multigrid algorithms. L1 and L3 cache measurements yield analogous results. Basically all required data are always found in the L1 and L2 cache. The combination of low cache misses with the data usage policy, i.e. one traversal per solver cycle and one data read per unknown, highlights that the present approach is memory modest. This statement holds independent of p. Our approach is not memory-bound though the grid changes almost each iteration and the code is a multiscale algorithm. It however neither exploits the available memory bandwidth which is around $8.35 \cdot 10^3$ MB/s for the Stream Triad [McCalpin 1995] benchmark ran on a single core with the same settings, nor does it exploit the vector registers. Its arithmetic intensity is very low. For $p > 3$, vectorisation even makes the runtime increase. We observe that few floating point operations per second face more than 10^{11} total instructions. The recursive code suffers from significant integer arithmetics; from administrative overhead.

We reiterate that (3) is to be solved for up to c channels simultaneously. As such, it is natural to make the solver tackle $\hat{c} \leq c$ problems simultaneously on the same grid. Each and every unknown associated to a vertex then is a $\mathbb{C}^{\hat{c}}$ tuple. Rather than relying on c independent problem solves, we fuse $\hat{c}$ problems into one setup solved on one grid. We refer to such an approach as *multichannel* variant. It is studied here at hands of five exemplary configurations (different mesh sizes, initial values, solvers) per p choice. Results from the Xeon Phi and Sandy Bridge qualitatively resemble each other though we have to take into account different constraints on the total memory—for $p = 4$, $h_{min} = 0.05$ already does not fit into the memory anymore for $\hat{c} > 8$.

We observe that solving multiple channels on one grid decreases the cost per unknown monotonously (Figure 14) up to $\hat{c} = 8$: the more channels fused into one grid the better the available memory bandwidth usage. Again, maintenance overhead is amortised. This holds despite the fact that the refinement criterion for the fusion of $\hat{c}$ channels into one grid has to be pessimistic. If one channel requires refinement, all $\hat{c}$ channels are mapped onto a finer grid. As our dynamically adaptive grid starts from few vertices and then refines, it successively amortises the overhead among the vertices for $p = 2$. The cost per vertex decreases. For $p = 3$ this effect is negligible. However, both p-choices exhibit cost peaks throughout the adaptive refinement due to additional initialisation effort. These relative cost are the smaller the more channels are fused.

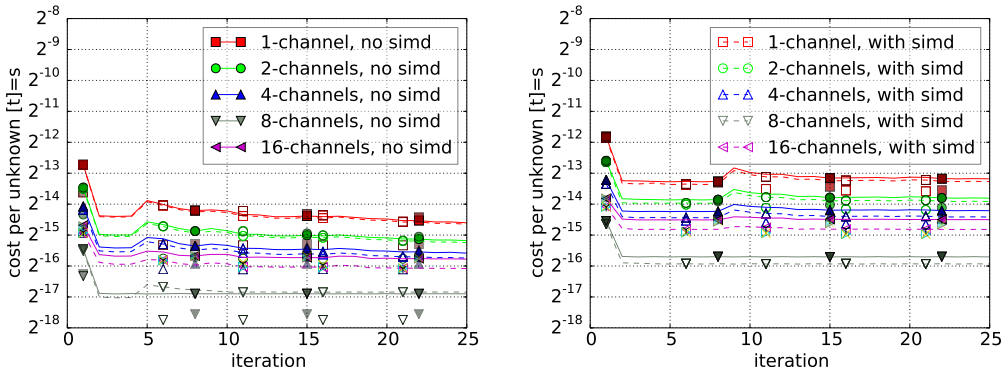

Fig. 14. Runtime per unknown per multigrid sweep for $p = 2$ (left) and $p = 3$ (right) on the Xeon Phi with several multichannel configurations and setups. Worst-case measurements with solid/dotted lines.

With bigger $\hat{c}$, the impact of vectorisation increases for $p = 3$ (as well as for $p = 4$ which is not shown here) while vectorisation is negligible or even counter-productive for $p = 2$. Even in the best case, it is still far below the theoretical upper bound. This is due to the fact that the multichannel approach does not change the arithmetic intensity of the compute kernels. All efficiency gains stem from an improved vectorisation as the kernels run through multiple channels per spacetree cell in a stream fashion.

We therefore propose to replace the Jacobi-like splitting of (2) by a block-structured decomposition $A \equiv \hat{A} + (A - \hat{A})$ with

$$\hat{A} = \begin{pmatrix} H_{11} & A_{12} & A_{13} & \dots & A_{1\hat{c}} & & & \\ A_{21} & H_{22} & A_{23} & \dots & A_{2\hat{c}} & & & \\ A_{31} & A_{32} & H_{33} & \dots & A_{3\hat{c}} & & & \\ \vdots & \vdots & \vdots & \ddots & \vdots & & & \\ A_{\hat{c}1} & A_{\hat{c}2} & \dots & \dots & H_{\hat{c}\hat{c}} & & & \\ & & & & & H_{(\hat{c}+1)(\hat{c}+1)} & \dots & A_{(\hat{c}+1)(2\hat{c})} \\ & & & & & \vdots & \ddots & \vdots \\ & & & & & A_{(2\hat{c})(\hat{c}+1)} & \dots & H_{(2\hat{c})(2\hat{c})} \\ & & & & & & & & \ddots \end{pmatrix}.$$

Though this is a stronger preconditioner and a harder system to solve, it retains the memory access requirements of the multichannel variant. However, such a *coupled multichannel* approach with its denser per-grid entity operators allows us to exploit vector facilities more efficiently since the arithmetic intensity increases.

An illustration of hardware counter measurements and timings validates and details these statements (Table IV). For small $1 \leq \hat{c} \leq 4$, the total runtime increases with an increase of $\hat{c}$, but the growth is sublinear. It is cheaper to compute multiple fused problems than to deploy them to grids of their own. For $4 \leq \hat{c} \leq 16$, the behaviour is non-uniform: While the runtime for some setups grows linear w.r.t. problems solved, it sometimes even drops. We deduce that this is the sweet region of an optimal choice of $\hat{c}$. The channel fusion really pays off if we fix $\hat{c} = 16$ ($p = 2$) or $\hat{c} = 8$ ($p \in \{3, 4\}$). For $\hat{c} \geq 16$, the runtime then grows linearly in the number of problems solved, i.e. the fusion does not pay off anymore. Furthermore, these large equation systems quickly exceed the main memory available on the Xeon Phi. Besides this constraint, the Phi's figures exhibit similar behaviour. For all setups, the bandwidth requirements do not

Table IV. Sandy Bridge performance for different coupled multichannel variants. One characteristic adaptive setting per choice of p. Vectorisation through Intel pragmas. Runtime is not normalised with problems solved simultaneously.

$p=2$	$\hat{c}=1$	$\hat{c}=2$	$\hat{c}=4$	$\hat{c}=8$	$\hat{c}=16$
Runtime	1.13e+02	1.67e+02	2.59e+02	5.20e+02	4.15e+02
MFlops/s	8.89e+02	1.91e+03	3.84e+03	6.85e+03	1.07e+04
L2 misses	2.77e-03	3.34e-03	4.33e-03	5.76e-03	7.19e-03
Used bandwidth (MB/s)	2.52e+02	3.15e+02	3.63e+02	3.51e+02	2.93e+02
$p=3$					
Runtime	2.11e+02	3.75e+00	4.56e+02	1.02e+03	2.78e+03
MFlops/s	1.40e+03	2.43e+03	6.02e+03	1.03e+04	1.44e+04
L2 misses	3.00e-03	3.46e-03	3.57e-03	4.42e-03	2.55e-02
Used bandwidth (MB/s)	1.37e+02	2.45e+02	1.78e+02	1.70e+02	1.32e+02
$p=4$					
Runtime	2.31e+03	3.39e+03	6.01e+03	3.07e+03	6.96e+03
MFlops/s	1.98e+03	3.87e+03	7.05e+03	1.08e+04	1.55e+04
L2 misses	8.29e-04	1.09e-03	3.63e-03	2.05e-02	3.01e-02
Used bandwidth (MB/s)	7.14e+01	7.78e+01	7.90e+01	1.71e+02	1.48e+02

increase significantly, and the cache misses remain negligible. While we cannot explain the drops in the execution times, we defer from the multichannel experiments that the runtime anomalies have to result from an advantageous usage of the floating point facilities—without a coupling, $\hat{c}=8$ is the sweet spot for all choices of p.

Besides the aforementioned amortisation of administrative overhead, the coupling with its increased matrices allows us to exhibit the vector facilities. We vectorise and reduce the relative idle times of the vector units compared to the total runtime. Both effects in combination improve the MFlop rate up to a factor of almost ten. This is around 20% of the theoretical peak though the measurements comprise all grid setup and management overhead. The depth-first traversal in the spacetree with its strictly local element operators allows us to achieve this without increased pressure on the memory subsystem. Due to the matrix-free rediscretisation approach, we can assume that the whole blocks of $\hat{A}$ reside within the cache. Due to the single touch policy per unknown and the localised traversal, we can assume that each unknown is loaded into the cache exactly once. This insight mirrors previous reports on this algorithmic paradigm [Mehl et al. 2006; Weinzierl 2009; Weinzierl and Mehl 2011]. We validate this at hands of the cache miss rate explaining the low bandwidth requirements.

Though the multichannel approach reduces the total concurrency, the channel block solves remain perfectly parallel. Due to the memory characteristics, we can expect that these block solves can run independently on the cores of our Sandy Bridge chip as one core does not consume more than $\frac{1}{16}$ of the available bandwidth. Due to the smaller main memory and the bigger core count, these multicore predictions do not apply unaltered to the Xeon Phi.

8. CONCLUSION AND OUTLOOK

The present paper introduces software and algorithms to realise a family of dynamically adaptive additive multigrid solvers for Poisson and Helmholtz problems working on complex-valued solutions. The latter provides a straightforward way to realise complex grid rotation. Supporting the approach's elegance and clarity, we provide correctness proofs, we demonstrate its robustness, and we discuss its runtime behaviour with respect to convergence speed and single-core utilisation. As the underlying algorithmic framework makes the solve of multiple Helmholtz problems or sets of problems perfectly parallel [Foster 1995], we focus on an algorithm that realises a single-touch policy—each piece of data is used only once or twice per traversal and the time in-between two usages is small—and strict localised operations. It does not stress the

memory subsystem. The experimental data reveals that this objective is met, while the solvers are reasonably robust and have a small memory footprint.

Besides an application to the underlying problems from physics and chemistry, we notably identify three methodological extensions of the present work. The first extension area tackles two obvious shortcomings of the present code base: smoothers and algebraic multigrid operators [Chen et al. 2012; Stolk 2015; Tsuji and Tuminaro 2015]. While work in this area is mandatory to facilitate the application of the algorithm and software idioms to more challenging setups, no fundamental risk, to the best of our knowledge, exists that these objectives can not be met. Several fitting building blocks are in place and have to be integrated properly. Our second extension area sketches open questions with respect to high-performance computing architectures. Finally, we pick up the problem of large p again.

Compared to previous work [Cools et al. 2014b], we have used a uniform complex grid rotation among all grid levels—well-aware that a level-dependent complex rotation might pay off. The same can be explored for the complex shifted ϕ approach from [Erlangga et al. 2004]. We reiterate that, for refined vertices, ϕ might hold both the sampled value plus an additional shift term and that, hence, no modification of any computation is required once ϕ is set—even for level-dependent complex shifts—since we determine $diag(H_\ell)^{-1}$ on-the-fly and do not rely on fixed diagonal values. Level-dependent shifts or grid rotations can either be determined a priori or whenever a simple operator analysis yields the insight that the operator enters a problematic regime (diagonal element underruns certain threshold or switches sign, e.g.). The latter property is particularly interesting for non-uniform ϕ distributions. Further next steps in the multigrid context comprise the realisation of operator-dependent grid transfer operators [Weinzierl 2013; Yavneh and Weinzierl 2012] and more suitable smoothers. Problem-dependent operators also might pay off along the boundary layers that are poorly handled by the current geometric operators. In this context, we emphasise that our algorithm relies on the Galerkin multigrid property in (7) while it so far realises rediscretisation. This introduces an error on the coarse grid. To the best of our knowledge, no analysis of such an error exists. Though one has to assume that it is bounded and small, explicit operator evaluation overcomes this problem completely. For future work tackling the smoother challenge, we refer in particular to patch-based approaches [Ghysels et al. 2013; Ghysels and Vanroose 2014]. Proof-of-concept studies from other application areas exist that use the same software infrastructure [Weinzierl et al. 2014] to embed small regular Cartesian grids into each spacetree cell. These small grids, patches, allow for improved robustness due to stronger smoothers resulting from Chebyshev iterations, higher-order smoothing schemes on embedded regular grids or a multilevel Krylov solver based on recursive coarse grid deflation [Sheikh et al. 2013; Erlangga and Nabben 2008]. Deflation is a particular interesting feature for highly heterogeneous Helmholtz problems where bound states emerge as isolated eigenvalues near the origin. These states are of special interest as they correspond to resonances in the system [Aguilar and Combes 1971; Balslev and Combes 1971; Moiseyev 1998; Simon 1979]. Their proper treatment seems to be implementationally straightforward within the present code idioms.

Patch-based approaches allow for efficient smoothers, but they are also promising with respect to multi- and manycores as well as MPI parallelisation. These two levels of parallelisation are a second track to follow. While our channels exhibit some perfect concurrency, the application of a proper domain decomposition on the long term will become mandatory due to the explosion of cores or high memory requirements. Several papers state that additive multigrid algorithms—though inferior to multiplicative variants in terms of convergence—are well-suited to parallel architectures due to their higher level of concurrency in-between the levels (notably [Chow et al. 2006] and

references therein or [Vassilevski and Yang 2014]). For the present codes, these statements are problematic. Our additive solvers exhibit a tight inter-grid data exchange and benefit from vertical integration. It is doubtful whether it is advantageous to deploy different grid resolution solves to different cores. Yet, the topic deserves further studies, notably if multiple sweeps are used to solve the individual levels' problems. Furthermore, the exact interplay of concurrent channel solves, time-in-between inter-grid data transfer and shared and distributed memory parallelisation remains to be investigated. Segmental refinement [Adams et al. 2016] in contrast seems to be an obviously promising technique. Starting from a reasonably fine grid, the underlying tree here is split up into independent subtrees deployed to different cores. This approach should integrate with the present ideas where we ensure high core efficiency through our vertical integration of various levels, while segmentation yields parallelism through horizontal decoupling of the grid levels.

To the best of our knowledge, $p \in \{3, 4\}$ in our application context already is a step forward compared to many state-of-the-art simulation runs. On the long term, however, bigger p have to be mastered. While stronger smoothers and better grid transfer operators might be able to deliver codes that do not suffer from the reduced coarse correction impact, the explosion of unknowns for $p \geq 5$ remains a challenge. Alternative techniques such as sparse grids [Bungartz and Griebel 2004] or sampling-based methods then might become a method of choice.

ACKNOWLEDGMENTS

Bram Reps is supported by the FP7/2007-2013 programme under grant agreement No 610741 (EXA2CT). Tobias Weinzierl appreciates the support from Intel through Durham's Intel Parallel Computing Centre (IPCC) which made it possible to access the latest Intel software. The AMR software base [Weinzierl et al. 2012] is supported and benefits from funding received from the European Unions Horizon 2020 research and innovation programme under grant agreement No 671698 (ExaHyPE). This work is dedicated to Christoph Zenger. He has first introduced the additive scheme in the context of Peano spacetree codes and he has acted as (co-)advisor to [Griebel 1990; 1994; Weinzierl 2009] which are starting points of the present work. All software is freely available from [Weinzierl et al. 2012].

APPENDIX

A. REMARKS ON TRADITIONAL ADDITIVE MULTIGRID

The top-down Algorithm 2 can be derived from the classic additive variant (Algorithm 1) in small steps. For this, we retain the bottom-up formulation (Algorithm 4) but, different to the plain correction scheme from Algorithm 1, already realise FAS.
This variant introduces a helper variable per level du_ℓ holding the impact of the smoother S, i.e. its correction of the solution on level ℓ. This helper variable translates into sc and sf for the final top-down algorithm. In the final line 11, we update the unknown on each level with a correction due to the smoother held in du_ℓ, and we also add the coarse grid contribution. To obtain a BPX-type solver, we have to rewrite this line into

$$u_\ell \leftarrow u_\ell + du_\ell + P\left(u_{\ell-1} - I(u_\ell + du_\ell)\right).$$

Both the coarse grid update and the smoother update are independent of each other and both rely on the injected solution from the previous traversal. Due to the former property, an additive scheme is realised. Due to the latter property, we may not only project some coarse data $u_{\ell-1}$, but we have to reduce this coarse contribution by the value injected from the current level Iu_ℓ. For the pure additive multigrid, this is one major difference to the similar scheme proposed in [Mehl et al. 2006] and references therein.

ALGORITHM 4: Additive FAS scheme running through the grid bottom-up, i.e. from fine grid to coarse level. Invoked by BUFAS(ℓ_{max}).

```
1:  function BUFAS(ℓ)
2:      du_ℓ ← ω_ℓ S(u_ℓ, b_ℓ)            ▷ Bookmark update due to a Jacobi step.
3:      if ℓ = ℓ_min then
4:          u_ℓ ← u_ℓ + du_ℓ
        else
5:          u_{ℓ-1} ← I u_ℓ                ▷ Inject into next coarser level
6:                                         ▷ to be able to compute the û_ℓ.
7:          û ← u_ℓ − P u_{ℓ-1}            ▷ Determine hierarchical surplus.
8:          r̂_ℓ ← b_ℓ − H_ℓ û
9:          b_{ℓ-1} ← R r̂_ℓ                ▷ Determine coarse grid rhs.
10:         BUFAS(ℓ − 1)                   ▷ Go to coarser level.
11:         u_ℓ ← u_ℓ + du_ℓ + P (u_{ℓ-1} − I u_ℓ)
        end
12: end function
```

LEMMA A.1. *The prolongation of $u_{\ell-1} - I(u_\ell + du_\ell)$ ensures that only a coarse correction is prolongated though the coarse grid holds a injected fine grid representation (FAS).*

PROOF. In what follows we will introduce the variable $corr_\ell = du_\ell - PIdu_\ell + Pcorr_{\ell-1}$ to label the update to the previous solution after one cycle. It consists of the net contribution $du_\ell - PIdu_\ell$ from the last smoothing step on the current level, plus the total prolongated correction from the coarse grid $Pcorr_{\ell-1}$. By definition, the new solution on level ℓ equals,

$$\begin{aligned}
u_\ell &= u_\ell^{prev} + du_\ell + P\left(u_{\ell-1} - I(u_\ell^{prev} + du_\ell)\right) \\
&= u_\ell^{prev} + du_\ell + Pu_{\ell-1} - PIu_\ell^{prev} - PIdu_\ell \\
&= u_\ell^{prev} + du_\ell + Pu_{\ell-1} - Pu_{\ell-1}^{prev} - PIdu_\ell \\
&= u_\ell^{prev} + du_\ell - PIdu_\ell + P(u_{\ell-1} - u_{\ell-1}^{prev}) \\
&= u_\ell^{prev} + du_\ell - PIdu_\ell + Pcorr_{\ell-1} \\
&= u_\ell^{prev} + corr_\ell.
\end{aligned}$$

We now show that the injection property holds, indeed for the finer level we have,

$$\begin{aligned}
Iu_{\ell+1} &= I(u_{\ell+1}^{prev} + corr_{\ell+1}) \\
&= Iu_{\ell+1}^{prev} + Idu_{\ell+1} - IPIdu_{\ell+1} + IPcorr_\ell \\
&= u_\ell^{prev} + Idu_{\ell+1} - Idu_{\ell+1} + IPcorr_\ell \\
&= u_\ell^{prev} + corr_\ell,
\end{aligned}$$

and so we can update the solution on level ℓ before the update on the finer level $\ell + 1$, and still conserve the injection property that is needed for the FAS scheme. □

The lemma's prolongation can be postponed to the subsequent tree traversal. It then acts as prelude there. This allows for the transform to a top-down algorithm and Algorithm 4 the logical starting point for the formulation of a top-down algorithm. We start the cycle with the update of the solution and the lines preceding the recursive call on line 10 are computed in a fine to coarse order. To facilitate this 'permutation', we introduced an additional variable $t_{\ell-1} = Idu_\ell$ that carries over the injected smoothing update from the previous cycle. Such a technique is a common pattern in pipelining.

B. REMARKS ON ELEMENT-WISE MATRIX-FREE MAT-VECS

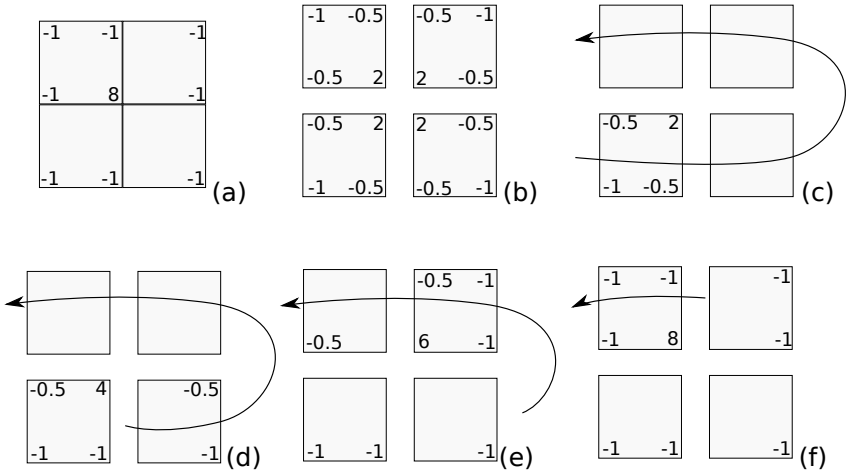

Fig. 15. Cartoon of element-wise matrix-free matrix-vector evaluations. (a) shows an exemplary stencil, i.e. one line of the matrix-vector product. (b) decomposes the stencil among the elements. These components are used within the cells, (c) through (f), that illustrate how the original stencil thus is successively reassembled. However, all data accesses are strictly element local.

The element-wise matrix-free evaluation of matrix-vector products (matvec) is a state-of-the-art technique in scientific computing. To simplify reproducibility and for matter of completeness, it is reiterated here.

We split up the stencils among the affected elements, i.e. rewrite them into an element-wise representation (Figure 15, (a) into (b)). For each vertex, we store a tuple with the current solution u as well as two helper variables r and $diag$.

The latter are set to zero before the traversal runs through any adjacent cell of the respective vertex, i.e. prior to Figure 15, (c). When we enter a cell, we read the p^2 adjacent u values and apply the local assembly matrix to these values. The result is added to the temporary variable r. Analogously, we accumulate the diagonal value $diag$. As we run through the grid, the whole stencil, i.e. assembly matrix line, is successively accumulated within r and $diag$ (Figure 15, (c) through (f)). Once all adjacent elements of a particular vertex have been run through, Figure 15 (f), r and $diag$ hold the matvec evaluation result associated to the vertex in r and the diagonal element value within $diag$. We add the right-hand side to r and update the u value using both r and $diag$.

REFERENCES

M. F. Adams, J. Brown, M. Knepley, and R. Samtaney. 2016. Segmental Refinement: A Multigrid Technique for Data Locality. *SIAM Journal on Scientific Computing* (2016). accepted.

J. Aguilar and J.M. Combes. 1971. A class of analytic perturbations for one-body Schrödinger Hamiltonians. *Communications in Mathematical Physics* 22 (1971), 269–279.

M. Bader. 2013. *Space-Filling Curves - An Introduction with Applications in Scientific Computing*. Texts in Computational Science and Engineering, Vol. 9. Springer-Verlag.

M. Baertschy and X. Li. 2001. Solution of a three-body problem in quantum mechanics using sparse linear algebra on parallel computers. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing*. ACM, 47–47.

E. Balslev and J.M. Combes. 1971. Spectral properties of many-body Schrödinger operators with dilatation-analytic interactions. *Communications in Mathematical Physics* 22 (1971), 280–294.

P. Bastian, W. Hackbusch, and G. Wittum. 1998. Additive and multiplicative multi-grid: a comparison. *Computing* 60, 4 (1998), 345–364.

A. Bayliss, C. I. Goldstein, and Eli Turkel. 1983. An iterative method for the Helmholtz equation. *J. Comput. Phys.* 49 (1983), 443–457.

A. Bayliss, C. I. Goldstein, and E. Turkel. 1985. On accuracy conditions for the numerical computation of waves. *J. Comput. Phys.* 59 (1985), 396–404.

R. Bellman. 1961. *Adaptive Control Processes: A Guided Tour*. Princeton University Press.

H. bin Zubair, S. P. MacLachlan, and C. W. Oosterlee. 2010. A geometric multigrid method based on L-shaped coarsening for PDEs on stretched grids. *Numerical Linear Algebra with Applications* 17 (2010), 871–894.

H. bin Zubair, B. Reps, and W. Vanroose. 2012. A preconditioned iterative solver for the scattering solutions of the Schrödinger equation. *Communications in Computational Physics* 11, 2 (2012), 415–434.

M. Bollhöfer, M.J. Grote, and O. Schenk. 2009. Algebraic multilevel preconditioner for the Helmholtz equation in heterogeneous media. *SIAM Journal on Scientific Computing* 31, 5 (2009), 3781–3805.

J. H. Bramble, J. E. Pasciak, and J. Xu. 1990. Parallel multilevel preconditioners. *Math. Comp.* 55 (1990), 1–22.

A. Brandt. 1973. Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. In *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics (Lecture Notes in Physics)*. 82–89.

A. Brandt. 1977. Multi-Level Adaptive Solutions to Boundary-Value Problems. *Math. Comp.* 31, 138 (1977), 333–390.

A. Brandt and I. Livshits. 1997. Wave-ray multigrid method for standing wave equations. *Electronic Transactions on Numerical Analysis* 6 (1997), 162–181.

H.-J. Bungartz and M. Griebel. 2004. Sparse grids. *Acta Numerica* 13 (2004), 147–269.

Z. Chen, D. Cheng, and T. Wu. 2012. A Dispersion Minimizing Finite Difference Scheme and Preconditioned Solver for the 3D Helmholtz Equation. *J. Comput. Phys.* 231, 24 (2012), 8152–8175.

E. Chow, R. D. Falgout, J. J. Hu, R. S. Tuminaro, and U. Meier Yang. 2006. A Survey of Parallelization Techniques for Multigrid Solvers. In *Parallel Processing For Scientific Computing*, M. Heroux, P. Raghavan, and H. Simon (Eds.).

A.T. Chronopoulos and C.W. Gear. 1989. s-step iterative methods for symmetric linear systems. *J. Comput. Appl. Math.* 25, 2 (1989), 153 – 168.

C. Cohen-Tannoudji, B. Diu, and F. Laloë. 1977. *Quantum Mechanics Volume 1*. Wiley-VCH.

S. Cools, B. Reps, and W. Vanroose. 2014a. An efficient multigrid calculation of the far field map for Helmholtz and Schrödinger equations. *SIAM Journal on Scientific Computing* 36 (2014), B367–B395.

S. Cools, B. Reps, and W. Vanroose. 2014b. A new level-dependent coarse grid correction scheme for indefinite Helmholtz problems. *Numerical Linear Algebra with Applications* 21 (2014), 513–533.

S. Cools and W. Vanroose. 2013. Local Fourier analysis of the complex shifted Laplacian preconditioner for Helmholtz problems. *Numerical Linear Algebra with Applications* 20, 4 (2013), 575–597.

J. Dongarra, J. Hittinger, and others. 2014. Applied Mathematics Research for Exascale Computing. (2014). DOE ASCR Exascale Mathematics Working Group: http://www.netlib.org/utk/people/JackDongarra/PAPERS/doe-exascale-math-report.pdf.

H.C. Elman, O.G. Ernst, and D.P. O'Leary. 2001. A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations. *SIAM Journal of Scientific Computing* 23, 4 (2001), 1291–1315.

B. Engquist and L. Ying. 2011. Sweeping Preconditioner for the Helmholtz equation: Moving Perfectly Matched Layers. *Multiscale Modeling & Simulation* 9 (2011), 686–710.

Y.A. Erlangga, C.W. Oosterlee, and C. Vuik. 2006. A novel multigrid based preconditioner for heterogeneous Helmholtz problems. *SIAM Journal on Scientific Computing* 27 (2006), 1471–1492.

Y. A. Erlangga and R. Nabben. 2008. On a multilevel Krylov method for the Helmholtz equation preconditioned by shifted Laplacian. *Electronic Transactions on Numerical Analysis* 31 (2008), 403–424.

Y. A. Erlangga, C. Vuik, and C. W. Oosterlee. 2004. On a class of preconditioners for solving the Helmholtz equation. *Applied Numerical Mathematics* 50 (2004), 409–425.

O. Ernst and M. Gander. 2011. Why it is Difficult to Solve Helmholtz Problems with Classical Iterative Methods, In Numerical Analysis of Multiscale Problems, I. Graham, T. Hou, O. Lakkis, and R. Scheichl (Eds.). *Numerical Analysis of Multiscale Problems* 83 (2011), 325–361.

L. D. Faddeev and S. P. Merkuriev. 1993. *Quantum Scattering Theory for Several Particle Systems*. Mathematical Physics and Applied Mathematics, Vol. 11. Springer Netherlands.

I. Foster. 1995. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc.

P. Ghysels, T. Ashby, K. Meerbergen, and W. Vanroose. 2013. Hiding global communication latency in the GMRES algorithm on massively parallel machines. *SIAM Journal on Scientific Computing* 35, 1 (2013), C48–C71.

P. Ghysels, P. Kosiewicz, and W. Vanroose. 2012. Improving the arithmetic intensity of multigrid with the help of polynomial smoothers. *Numerical Linear Algebra with Applications* 19, 2 (2012), 253–267.

P. Ghysels and W. Vanroose. 2014. Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm. *Parallel Comput.* 40, 7 (2014), 224 – 238. 7th Workshop on Parallel Matrix Algorithms and Applications.

P. Ghysels and W. Vanroose. 2015. Modeling the performance of geometric multigrid on many-core computer architectures. *SIAM Journal on Scientific Computing* 37, 2 (2015), C194–C216.

M. Griebel. 1990. *Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hiearchischen-Transformations-Mehrgitter-Methode*. Vol. 342/4/90 A. Dissertation, Technische Universität München.

M. Griebel. 1994. Multilevel Algorithms Considered As Iterative Methods On Semidefinite Systems. *SIAM Journal on Scientific Computing* 15, 3 (1994), 547–565.

E. Haber and S. MacLachlan. 2011. A fast method for the solution of the Helmholtz equation. *J. Comput. Phys.* 230 (2011), 4403–4418.

G. Ronald Hadley. 2005. A Complex Jacobi Iterative Method for the Indefinite Helmholtz Equation. *J. Comput. Phys.* 203, 1 (Feb. 2005), 358–370. DOI:http://dx.doi.org/10.1016/j.jcp.2004.09.015

M. Hoemmen. 2010. *Communication-avoiding Krylov subspace methods*. Ph.D. Dissertation. EECS Dept., U.C. Berkeley.

D.A. Horner, F. Morales, T.N. Rescigno, F. Martín, and C.W. McCurdy. 2007. Two-photon double ionization of helium above and below the threshold for sequential ionization. *Physical Review A* 76, 3 (2007), 30701.

F. Ihlenburg and I. Babuska. 1995. Finite element solution to the Helmholtz equation with high wave numbers. *Computers and Mathematics with Applications* 30 (1995), 9–37.

M. Kowarschik, U. Rüde, C. Weiß, and W. Karl. 2000. Cache-Aware Multigrid Methods for Solving Poisson's Equation in Two Dimensions. *Computing* 64, 4 (2000), 381–399.

M. Magolu monga Made, R. Beauwens, and G. Warze. 2000. Preconditioning of discrete Helmholtz operators perturbed by a diagonal complex matrix. *Communications in Numerical Methods in Engineering* 16, 11 (2000), 801–817.

J.D. McCalpin. 1995. Memory Bandwidth and Machine Balance in Current High Performance Computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter* (Dec. 1995), 19–25.

K. Meerbergen and J.P. Coyette. 2009. Connection and comparison between frequency shift time integration and a spectral transformation preconditioner. *Numerical Linear Algebra with Applications* 16, 1 (2009).

M. Mehl, T. Weinzierl, and C. Zenger. 2006. A cache-oblivious self-adaptive full multigrid method. *Numerical Linear Algebra with Applications* 13, 2–3 (2006), 275–291.

N. Moiseyev. 1998. Quantum theory of resonances: calculating energies, widths and cross-sections by complex scaling. *Physics Reports* 302 (1998), 211.

D. Osei-Kuffuor and Y. Saad. 2010. Preconditioning Helmholtz linear systems. *Applied numerical mathematics* 60, 4 (2010), 420–431.

National Research Council Plasma 2010 Committe, Plasma Science Committee. 2007. *Plasma Science: Advancing Knowledge in the National Interest*. The National Academies Press.

R.E. Plessix and W.A. Mulder. 2003. Separation-of-variables as a preconditioner for an iterative Helmholtz solver. *Applied Numerical Mathematics* 44 (2003), 385–400.

Y. P. Raizer. 1991. *Gas Discharge Physics*. Springer-Verlag Berlin Heidelberg.

J. Reinders and J. Jeffers. 2015. *High Performance Parallelism Pearls*. Elsevier.

B. Reps, W. Vanroose, and H. bin Zubair. 2010. On the indefinite Helmholtz equation: Complex stretched absorbing boundary layers, iterative analysis, and preconditioning. *J. Comput. Phys.* 229, 22 (2010), 8384–8405.

E. Schrödinger. 1926. An unulatory theory of the mechanics of atoms and molecules. *Physical Review* 28 (1926), 1049–1070.

A. H. Sheikh, D. Lahaye, and C. Vuik. 2013. On the convergence of shifted Laplace preconditioner combined with multilevel deflation. *Numerical Linear Algebra with Applications* 20 (2013), 645–662.

B. Simon. 1979. The definition of molecular resonance curves by the method of exterior complex scaling. *Physics Letters A* 71 (1979), 211.

B. M. Smirnov. 2015. *Theory of Gas Discharge Plasma*. Springer Series on Atomic, Optical, and Plasma Physics, Vol. 84. Springer International Publishing.

C. C. Stolk. 2015. *A dispersion minimizing scheme for the 3-D Helmholtz equation with applications in multigrid based solvers*. Technical Report arXiv:1504.01609.

H. Sundar, R. S. Sampath, and G. Biros. 2008. Bottom-Up Construction and 2:1 Balance Refinement of Linear Octrees in Parallel. *SIAM Journal on Scientific Computing* 30, 5 (2008), 2675–2708.

J. Treibig, G. Hager, and G. Wellein. 2010. LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments. In *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops (ICPPW '10)*. IEEE Computer Society, 207–216.

U. Trottenberg, Oosterlee C. W., and A. Schuller. 2001. *Multigrid*. Academic Press, Inc.

P. Tsuji and R. Tuminaro. 2015. Augmented AMG-shifted Laplacian preconditioners for indefinite Helmholtz problems. *Numerical Linear Algebra with Applications* 22, 6 (2015), 1077–1101.

M.B. van Gijzen, Y.A. Erlangga, and C. Vuik. 2007. Spectral Analysis of the Discrete Helmholtz Operator Preconditioned with a Shifted Laplacian. *SIAM Journal on Scientific Computing* 29, 5 (2007), 1942–1958.

W. Vanroose, F. Martin, T. N. Rescigno, and C. W. McCurdy. 2005. Complete photo-induced breakup of the H_2 molecule as a probe of molecular electron correlation. *Science* 310 (2005), 1787–1789.

P. S. Vassilevski and U. Meier Yang. 2014. Reducing communication in algebraic multigrid using additive variants. *Numerical Linear Algebra with Applications—Special Issue: Multigrid methods 2013* 21, 2 (2014), 275–296.

M. Weinzierl. 2013. *Hybrid Geometric-Algebraic Matrix-Free Multigrid on Spacetrees*. Dissertation. Fakultät für Informatik, Technische Universität München, München.

T. Weinzierl. 2009. *A Framework for Parallel PDE Solvers on Multiscale Adaptive Cartesian Grids*. Verlag Dr. Hut, München. http://www.dr.hut-verlag.de/978-3-86853-146-6.html

T. Weinzierl. 2015. *The Peano software—parallel, automaton-based, dynamically adaptive grid traversals*. Technical Report 2015arXiv150604496W.

T. Weinzierl and others. 2012. Peano—a Framework for PDE Solvers on Spacetree Grids. (2012). http://www.peano-framework.org

T. Weinzierl, M. Bader, K. Unterweger, and R. Wittmann. 2014. Block fusion on dynamically adaptive spacetree grids for shallow water waves. *Parallel Processing Letters* 24, 3 (2014), 1441006.

T. Weinzierl and M. Mehl. 2011. Peano – A Traversal and Storage Scheme for Octree-Like Adaptive Cartesian Multiscale Grids. *SIAM Journal on Scientific Computing* 33, 5 (2011), 2732–2760.

I. Yavneh and M. Weinzierl. 2012. Nonsymmetric Black Box multigrid with coarsening by three. *Numerical Linear Algebra with Applications* 19, 2 (2012), 246–262.

Received t.b.d.; revised t.b.d.; accepted t.b.d.